1

AN EXPLORATORY APPLICATION OF NEURAL NETWORKS TO THE SORTIE GENERATION FORECASTING PROBLEM

THESIS

James M. Dagg, GS-12

AFIT/GLM/LSM/91S-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GLM/LSM/91S-11

AN EXPLORATORY APPLICATION OF NEURAL NETWORKS TO THE SORTIE GENERATION FORECASTING PROBLEM

THESIS

James M. Dagg, GS-12

AFIT/GLM/LSM/91S-11

AFIT/GLM/LSM/91S-11

AN EXPLORATORY APPLICATION OF NEURAL NETWORKS TO THE SORTIE GENERATION FORECASTING PROBLEM

THESIS

Presented to the Faculty of the School of Systems and
Logistics of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Logistics Management

James M. Dagg, G.A.
GS-12

September 1991

Preface

This follow-on study investigates specific aspects of F-16 air base simulation modeling and analysis originally discussed in the much broader research of Dr. David A. Diener, Major, USAF. As the initial research increment to be improved upon by future researchers, this study (1) provides a preliminary assessment of the relative accuracy of conservatively specified neural network metamodels in predicting sortie generations for pre-specified F-16 air base resource level postures, and (2) presents a possible research methodology for future investigations.

Backpropagation neural networks (single hidden layer) and regression equations are fitted to daily cross-sections of simulated data representing the first six days of thirty day (longitudinal) time series data previously generated *via* simulation by Diener. Two scenarios are modeled, Attack and No-Attack, yielding a total of six network metamodels and six associated regression metamodels (used as a baseline for predictive accuracy) per scenario. An independent test sample not used for metamodel fitting is generated for each day, and the predictive accuracy of each daily metamodel is evaluated. For this study, the conclusion is negative: regression metamodels are found superior in predicting unseen cases comprising the independent test samples, in comparison with their neural network counterparts.

Given the severity of the experimental constraints, the fact that all samples represent independent variables as dichotomous values, the relatively small size of the sample (from the perspective of neural network training requirements), and the robust nature of traditional regression modeling, network results are promising. The networks exhibit rather remarkable memorization (goodness-of-fit) characteristics in spite of their seeming inability to accurately generalize (predict) well for unseen cases. It is suggested that the lack of generalization exhibited by the networks developed in this study can be remedied (for Attack Scenario days in particular) *via* modifying network architectures to account for blocking, increasing training set (sample) sizes, and/or employing a hybrid net for modeling (such as Kohonen-Backpropagation).

This work owes its genesis to the efforts of several individuals providing support and assistance at precisely the right moments throughout the research process. First, I wish to thank David A. Diener, Major, USAF, who virtually transformed my dream of exploring neural network techniques into concrete reality. His talents in simulation, modeling, and statistical analysis are surpassed only by his tolerance, patience, kindness, and a remarkably steadfast sense of purpose. Second, I owe very deep gratitude to Jerry Galligher of California Scientific Software, whose willingness and ability to answer any question I had about network theory and applications speaks as much for his humanity as for his intellectual capabilities. Third, thanks is owed to Jacob Simons, Major, USAF, for patiently and consistently reminding me that the entire point of research such as mine is ultimately to assist people in making difficult decisions about real-world problems. Fourth, I wish to thank Dr. Ken Melendez, Deputy Director of the Center for Artificial Intelligence Applications (CAIA), for advice and recommendations that proved to be critical during the beginning stages of this research. Fifth, a word of thanks is also owed to Dr. P. George Benson (Curtis L. Carlson School of Management, University of Minnesota), Dr. Shane, Dr. Barr, and Dr. Donna Herge, Lt Col, USAF, all AFIT faculty members, for guidance regarding appropriate statistical analysis and research methodology techniques. Finally, I wish to express deep gratitude to my family--to my wife Mary, for the understanding, concern, and inspiration she so selflessly gave me throughout the course of this work, and to Pierrette and Jim, for being wise far beyond their years in somehow understanding why their father was always so very busy.

Table of Contents

List of Figures

List of Tables

AFIT/GLM/LSM/91S-11

Abstract

This exploratory study assesses the accuracy of backpropagation neural networks in predicting sortie generations, given pre-specified levels of air base resources. Single hidden layer networks and two-way interaction regression metamodels were fitted to simulated data previously generated by way of a fractional factorial design for ten factors at two levels, and subsequently tested (cross-validated) via an independent testing sample. It was determined that regression metamodels were generally superior in predicting unseen cases, while their network counterparts exhibited far better goodness-of-fit characteristics.

The research consistently emphasizes that goodness-of-fit in no way necessarily implies goodness-of-prediction, in that different non-equivalent statistical measures are required to assess both these phenomena.

In spite of their relatively poor performance in predicting the test sample used in this study, experimental results indicate that future research focused on applying neural network modeling techniques to sortie generation prediction and the identification of critical air base resources is warranted. Suggestions for more effective neuronal modeling include architectural experimentation, increasing sample sizes employed for network training, and representing the research problem in terms of a time series.

AN EXPLORATORY APPLICATION OF NEURAL NETWORKS TO THE SORTIE GENERATION FORECASTING PROBLEM

I. Introduction

1.1 Air Base Operability (ABO) System Performance

1.1.1 Resource Management.

As identified by Tidal McCoy, former Assistant Secretary of the Air Force, ABO consists of four primary functions: defense, survival, recovery, and continuing to fly aircraft (1987:52-56). Insightful resource management is central to this paradigm in that operational effectiveness depends upon continuing logistics infrastructure health. Understanding the relationship between resource allocation decisions and ABO performance is therefore critical for managers attempting to optimally distribute scarce financial and physical resources among Air Base functional elements.

Choosing from the array of resource allocation alternatives requires the perspective of what is most beneficial (optimal) for overall ABO *system performance*. Managers must perform this selection within a highly dynamic environment characterized by numerous mission goals, resource shortages, time constraints, hostile encounters, and other complexities. The emerging requirement is thus effective *systems-level* resource management, with a corequisite need for support systems and analysis tools that foster informed decision making within a highly dynamic decision environment.

1.1.2 Large-Scale ABO Systems Modeling. The RAND Corporation has developed several analytic and simulation models of combat support systems for the research of complex system interactions. One simulation model set, the Theater Simulation of Air Base Resources (TSAR) and its companion model, TSAR Inputs using AIDA (TSARINA), provides a powerful means for evaluating alternative systems-level resource policy scenarios for several Air Base system configurations. This simulation provides data necessary for analyzing ABO system performance by generating sorties flown and other operational measures amenable to statistical analysis and interpretation (Rich et al., 1987).

ABO modeling and analysis are foundational for accurately assessing readiness and sustainability, for examining the support performance characteristics of an existing logistics infrastructure (the ABO support system), and for (ultimately) providing resource managers with the information required to optimally allocate resources to Air Base functional elements. While several other models are available to ABO researchers, this research discusses only the TSAR/TSARINA model set and the data it provides for analysis.

1.1.3 Definitions. Given a requirement for a systems-level approach for ABO analysis, a clear understanding of the terms *sortie generation*, *resources*, and *sorties generated* (or *sorties flown*) is required. In this research, *sortie generation* is the process by which aircraft are made ready to fly. The inputs to this process are the existing *resources* an air base possesses, while the outputs of this process are the resulting number of aircraft flights--*sortie generations* or *sorties flown*. A sortie is to be understood as one aircraft *flight*. The number of sorties flown are often characterized temporally, (as the number of flights occurring within a given time interval), in order to provide a basis for comparison (for example, the number of flights occurring within a day, tabulated for 30 days). For this research, the terms *sorties generated* and *sorties flown* are considered equivalent (both denote the number of flights occurring within a specified time interval); both are used interchangeably throughout this research.

The relationship between resources and sortie generations (or sorties flown), denoted by *sortie generation*, is exceedingly complex and dynamic in nature. Diener researches this process, illustrated in his study and duplicated here as Figure 1.1, in terms of a mathematical (functional) relationship (1989:12). Resources are characterized as independent variables and sorties flown as the response. Note that the sortie generation process is conceived as dynamic in that changes in resource levels (resource level dynamics) are assumed to produce significant differences in the numbers of sorties flown, and also because the *interaction* of resources is assumed to be highly significant (Diener, 1989:11-15). These dynamics are further increased when the impacts of air base attacks on resource levels are considered. The process, inputs, and outputs are collectively represented in Diener and duplicated here as a companion to Figure 1.1, in Figure 1.2 (1989:13).

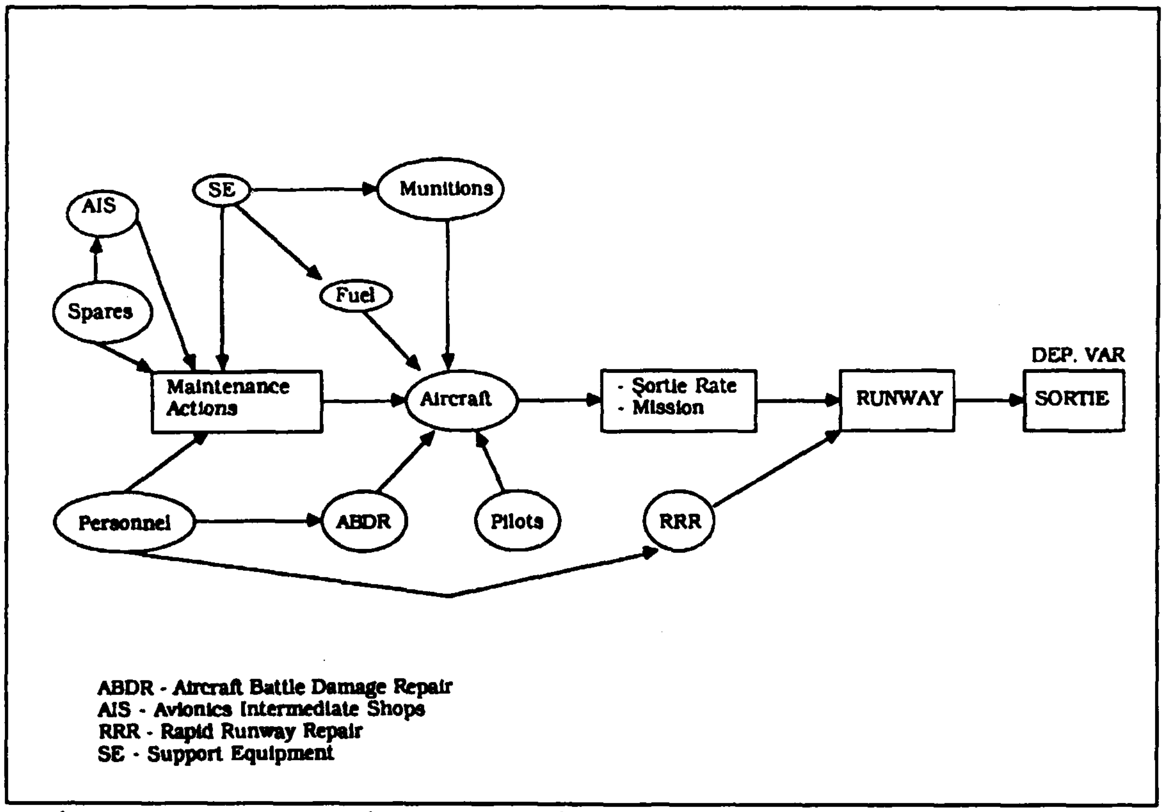


Figure 1.1. Sortie Generation Process (Diener, 1989:12)

Finally the term *scenario*, as used in this study, denotes an ABO environmental variable that may assume one of two possible values--*attack* or *no-attack*. In the case of *attack*, air base resources are potentially lost or destroyed. No continuum of range is considered with respect to the value of the variable *attack*.

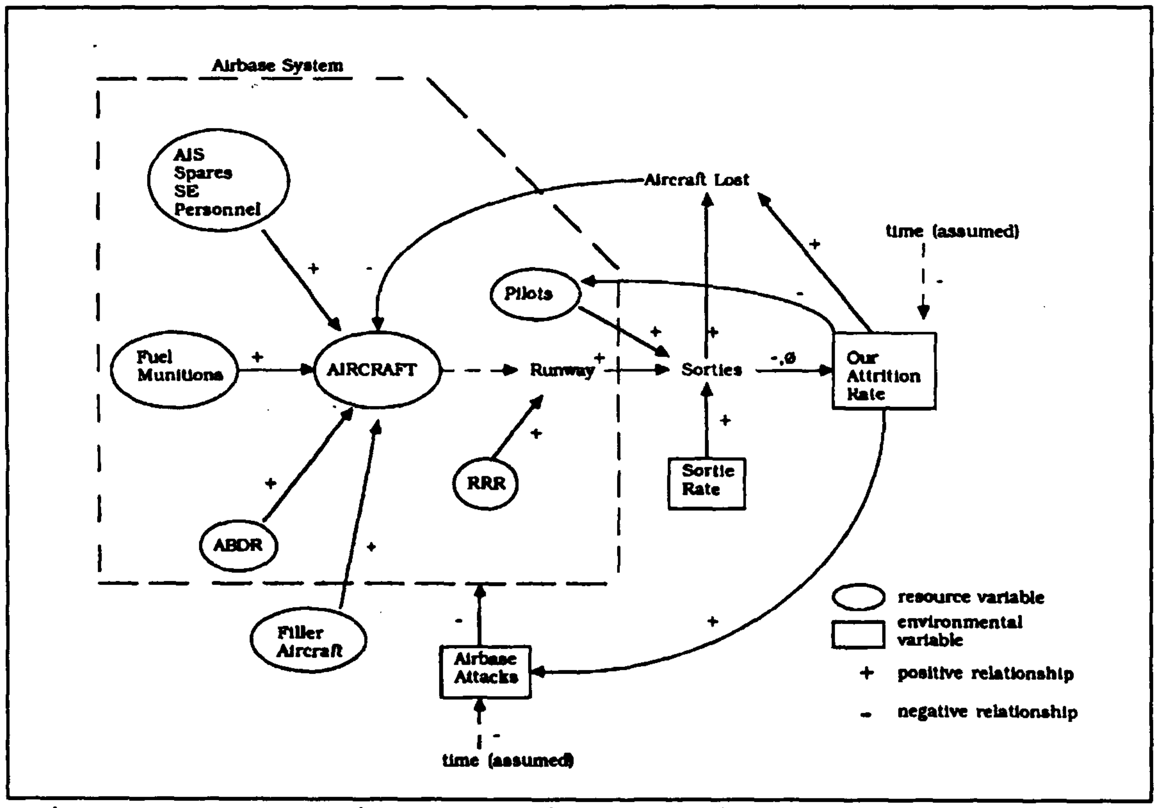


Figure 1.2. Sortie Generation Dynamics (Diener, 1989:13)

1.1.4 <u>ABO Analysis Approach</u>. Interpreting the McCoy paradigm referenced above in operational terms, the investigation of the relationship between resource dynamics and sorties flown is a logical modeling candidate and starting point for analyzing the impacts of resource alternatives on ABO system performance (Diener, 1989:1-4). While the TSAR/TSARINA simulation model provides necessary data, a well-defined analysis methodology and logical interpretation paradigm for analysis results are obviously required. In the following discussion of the general problem for this study--the modeling and analysis of the relationship between sortie generations and resource dynamics--both the system definition (one F-15 Air Base) and general approach to analysis previously employed by Diener are noted and adopted (1989). The conception of the relationship between resources and sortie generations as a *function* is considered fundamental.

1.2 General Research Problem and Goal

1.2.1 Previous Study. In 1989, Diener employed a systems-level modeling perspective for analyzing several ABO research issues (1989). Two elements of the stated research objectives were predicting daily sortie generations and identifying critical resource changes and interactions occurring throughout a one month simulation analysis horizon (1989:5). All ABO resource, environmental, and policy variables were conceptualized as a set of mutually interrelated and co-dependent elements that generally defined the air base logistics infrastructure--whose system performance was measured in terms of sortie generations.

An important development in this study was the use of regression models for both predicting sortie generations and analyzing resource dynamics (1989:6-9). Through modeling an F-15 Air Base first as a system *via* large-scale simulation (TSAR/TSARINA), regression equations were subsequently derived from generated data as more general analytic *meta-models* of the simulation model itself. Analyses and interpretations of resource level main effects and secondary interactions were performed *via* these metamodels, for the purpose of evaluating the behavior of resource dynamics occurring throughout the one month simulation horizon.

Sortie generation prediction and resource level dynamics analyses were accomplished for various resource policy scenarios in both the attack and no-attack scenarios. The analytic metamodels were then offered as an economical alternative to large-scale simulation, and, further, as a methodology ultimately intended to assist resource managers in assessing the impact of policy or resource allocation alternatives on ABO system readiness and sustainability support characteristics.

1.2.2 Present Study Research Problem and Goal. Thus, the analytical modeling and analysis of ABO system sortie generation and resource interaction dynamics provide the general research domain of this follow-on study. The overall goal of this research is to derive, if possible, more accurate analytical metamodels for predicting sortie generations and interpreting resource dynamics than those previously developed using linear techniques. This is attempted *via* the use of a specific non-linear modeling technique, with emphasis placed on the potential value of an analytic alternative to the time and cost intensive technique of large-scale simulation. Prior to discussing the specifics of this technique and the approach to the problem, the general issues of the present research environment, largely inherited from the progenitor study referenced above, are presented next.

1.3 Research Issues Inherited from Diener

Diener conducted the TSAR ABO simulation as a controlled statistical experiment, utilizing a fractional factorial experimental design for treatment selection and application (1989). Through the use of this design together with the employment of a common random number (CRN) variance reduction technique (VRT), the resolution of essential factor treatment/response data was maximized throughout the attempt to reduce the impact of simulation model variance on experimental results. Metamodel parameter estimates were subsequently developed using the data sample generated by the simulation. Finally, the derived metamodels were used for measuring treatment (resource level combination) effects on the system response variable (sorties flown), and for interpreting resource dynamics (main effects and second order interactions). With respect to the metamodels, the four key issues requiring examination are model linearity, variance, response correlation, and prediction versus fit.

1.3.1 Linearity. Throughout Diener's study, it was assumed that the relationship between daily sorties flown and resource levels was linear, and that higher order interaction terms were negligible (an assumption often made by researchers using two-level factorial designs for exploratory data analysis) (Diener, 1989:42; McLean and Anderson, 1984:1). Neither assumption may be entirely without difficulty (Meyers, 1986:167-168; Diener, 1989:221). While the experimental design employed was sufficient for preventing the occurrence of confounding between main effects and secondary interactions within the statistical experiment, it in no way conclusively demonstrates or provides guarantees for the assumed linear relationship.

Assuming linearity requires the researcher to make assumptions about the random error term population distribution, its mean value, the constancy of its variance, and the independence of the random errors themselves (McClave and Benson, 1988:500-503,557-558). If the assumptions of linearity do not obtain (are not evident in the data), both forecasting accuracy and resource analysis results may be potentially compromised. Non-linear models may be more appropriate and may provide more accurate predictions than their linear counterparts, particularly when assumptions of linearity are violated--when a linear relationship is not truly present (Meyers, 1984:300-302,312-316).

1.3.2 Variance. In regard to unexplained variance, Diener states:

> The air base logistics infrastructure as modeled here still contains much unexplained variance as evidenced by the R^2 results. The average R^2 for the attack case is 0.6366 while the no-attack case averages 0.6789. Either the problem has a high degree of inherent variability, some other important factors are omitted, or some higher-interaction terms are not negligible. (1989:221)

Although much effort was made to diminish the effects of simulation model variance by employing a CRN technique based on common random number stream starting points (seeds) across design points (runs), a large degree of variance remains unexplained. Another issue, then, entails identifying both the sources of this variance and their respective contributions. At a more general level, a method of effectively dealing with the impact of variance on prediction accuracy and resource dynamics interpretation are the primary concerns.

1.3.3 Correlation. Each run of the simulation model encompasses a thirty day span, where the treatment specified represents the state of the world as of Day 1:

> Thus we are dealing with indicator variables representing logistics policies, repair capabilities, expected delivery schedules, and resource levels as of Day 1, rather than trackable quantitative series for the independent variables. (Diener, 1989:15)

The issue here involves correlated responses (daily sorties flown) *within* the time series of 30 daily sorties flown measures generated by each run (Diener, 1989:14-15,46). While the impact of correlation is mitigated by cross-sectional metamodel derivation *across* runs (one metamodel is developed for each day--across treatments), potential problems of autocorrelation, nonstationarity, and series interruptions (resulting from the shocks of TSARINA attack simulations) may arise when evaluating the response (sorties flown) time series generated within a run (Diener, 1989:14-15). The effects of (temporal) correlation on the experimental results constitutes the third research environment issue inherited by this study.

1.3.4 Model Fit versus Predictive Accuracy. Research suggests that the comparison of R^2 values and backward step-wise elimination methodologies (based on α significance level comparisons) often used for linear model selection (Meyers, 1986:55,219) may not yield the best candidates for use in *prediction*, but are rather more appropriate for assessing the best overall model *fit* (Meyers, 1986:100-111). Thus, a well-defined distinction between model fit and predictive accuracy is required.

In his discussion of criteria for model selection, Meyers notes the following regarding the coefficient of determination, R^2, of a model:

> As we indicated earlier, R^2 is surely a measure of the model's capability to fit the present data. ... Though there are rules and algorithms (strictly based on R^2) that allow for selection of best model, the statistic itself is not conceptually prediction oriented (i.e., prediction performance based); thus it is not recom-

mended as a sole criteria for choosing the best prediction model from a set of candidates. (1986:102)

Further, with respect to the analysis of residuals, he notes: "These residuals are measures of the quality of fit and *do not assess the quality of future predictions*" (1986 :103). The point is that in model selection, one must not only take into account the fit of a model to the present data, but must also contend with the issue of predictive accuracy--a related, though distinct, issue.

Note that this does not suggest that models exhibiting large R^2 values are inappropriate--such models, judiciously selected *via* this and other diagnostic measures, are fundamental to data analysis in regard to understanding the system under consideration. What is suggested is that a true difference between understanding (with respect to fitting) a system and predicting future values of same does in fact exist, and that both issues must be considered by the researcher during the model selection process. This difference constitutes the final consideration necessary for assessing the metamodels previously developed by Diener (1989).

1.4 Specific Research Problem and Approach

An empirical investigation is conducted to determine whether a certain class of non-linear models will yield consistently more accurate sortie forecasts than is possible using the traditional approach employed by Diener (1989).

Assuming the simulation output data represent real-world conditions, twelve feedforward backpropagation neural networks are developed (with software) to model each of the first six (of thirty) days of the air base attack and non-attack scenarios (the first six days of the attack scenario have proven to be the "messiest" time period in the model and are intentionally chosen for analysis).

The feedforward/backpropagation network model combination has been specifically selected for testing for several reasons, including its ability to model arbitrarily complex decision surfaces effectively, its freedom from dependency on linear superposition and orthogonal functions, its theoretical ability to model any mathematical function, and its widespread use in applications (Hecht-Nielsen, 1990:94, 108; Hecht-Nielsen, 1988:120-121; Lippman, 1987:50; Wasserman, 1989:43-60). Neural networks provide a new technique for exploring alternative approaches in ABO modeling research.

1.5 Research Objectives

The research objectives germane to this follow-on study focus specifically on the application of neuronal modeling to the sortie forecasting problem and the interpretation of resource level dynamics. The study is organized in terms of the following objectives:

1. Feedforward backpropagation neural networks are trained and tested to determine whether they can provide

better predictions of sorties flown (for a specified set of days) than linear regression equations. All models are developed (fitted) from the sample data previously generated and analyzed in Diener (1989).

2. All neural network and regression equation metamodels are tested on an independent sample not used in model fitting, to specifically determine how well each method predicts the simulated response variable (sorties flown). The decision to model the "difficult" days from Diener's study (the first six of both scenarios) is based on the desire to assess predictive accuracy. An associated issue with regard to this objective lies in clearly distinguishing predictive capability from goodness-of-fit, and assessing, to the extent possible, how they are or are not related.

3. An interpretation of the neural network model performance will be provided, stated in terms of specific formal and/or logical properties. In addition, the inherited research issues of linearity, correlation, and variance are addressed in this context.

1.6 <u>Limitation of Scope</u>

The wealth of available ABO modeling and analysis literature is clearly indicative of its breadth, complexity, and diversity. In the interests of analytical clarity and focus, this research is strictly limited to developing and analyzing twelve feedforward neural network models, (using supervised learning and the backpropagation training rule), which model each of the first six days of the attack and no-attack scenarios presented in Diener's work (1989), *via* comparison with corresponding linear regression model counterparts.

Data provided by the TSAR/TSARINA model set are assumed to be generated by processes completely isomorphic to real-world processes and representative of system state treatment responses of the F-15 Air Force Base envisioned by Diener.

The network models developed herein are to be viewed as the product of a methodology to be used for further exploratory research and are not intended for generalized use, particularly in field environments.

1.7 Analysis Plan

Chapter II begins by providing definitions of readiness and sustainability, and their relationship within the ABO modeling environment. Section 2 describes the statistical experiment used by Diener for data generation and subsequent analysis. In Section 3, the concept of metamodeling is defined, and its relationship to experimental design, its derivation process, and its importance in applied research is discussed. Section 4 provides an historical overview of theoretical neural network concepts germane to this study. Finally, Section 5 reviews selected neural network research literature.

Section 1 of Chapter III describes the sample used for metamodel development and discusses the implications of using simulation samples in lieu of real-world data for model fitting and training. Section 2 discusses the regression model methodology used for this study. Section 3 presents methodological considerations in regard to building, training, and testing the neural network models. Finally, Section 4 lists the measures used to assess comparative predictive accuracy, while Section 5 discusses recognized modeling limitations and comments on comparative model relationships.

Section 1 of Chapter IV presents the analysis of neural network metamodel architectural formulation and training. Section 2 discusses the analysis and findings regarding regression/network metamodeling fitting, while Section 3 presents the comparative analysis of predictive accuracy for both metamodel types. Finally Section 4 of this chapter explores the extent to which metamodel forecasts are statistically significantly different from sample actuals, and Section 5 summarizes principal findings.

Chapter V, the concluding chapter, discusses how metamodel developments and analyses directly support the stated research objectives, provides overall conclusions for the thesis, and ends with recommendations for future research.

II. Background and Literature Review

2.1 Modeling Combat Capability

2.1.1 Systems Perspective. In their discussion of assessing readiness and sustainability in the context of combat capability, RAND researchers Rich et al. note:

> Readiness and sustainability complement the other two components of combat capability: force structure and force modernization. Readiness is conceived of as the force's ability to execute its combat mission effectively with little notice; sustainability is the force's ability to pursue that mission for a long period of time. (1987:2)

In simple terms, an approach to measuring readiness may be attempted, in part, by observing the elapsed time between the initiation of hostility and an effective response to same, while sustainability may be addressed by analyzing the extent to which a system's performance does or does not degrade throughout long-term engagement. In broader terms, both concepts share several analysis dimensions, including the quantity and quality of system responses, the extent to which the system can provide continued mission support, and the extent to which the system can withstand attack damage and recover. More importantly, these two combat capability components are temporal (both must be measured in terms of time), mutually interact, and are critically related to the same underlying support system--the logistics infrastructure. This research focuses on the support system, and attempts to understand Diener's research in *explaining* the effects of logistics infrastructure dynamics on sortie generation, and to explore an alternative method for *predicting* sortie generations based on different resource level policies. To facilitate analysis, a unified analytical framework is clearly indicated for attempting to discover significant characteristics of highly dynamic behavior of logistics support systems within combat environments.

As a starting point, a systems perspective is adopted, and research issues are explored as they relate to overall system performance. In this research, the system under consideration is a single F-15 air base, where inputs are transformed to effect the realization of responses to hostilities (resources are transformed *via* the sortie generation process into sorties flown). Because the combat capability components in question both share a temporal dimension and a common logistics infrastructure (the air base as a system), system outputs (sorties flown) must be measured temporally in terms of a time series. In addition, the inputs to the process must be clearly identified to allow accurate analysis; in this study, inputs consist of 9 resources, and the environmental variables of aircraft attrition and attack/no-attack scenario. Further, the impact of attacks on both the system and its resources must be considered for any realistic assessment of air base operability. With these postulations of the system, its inputs, outputs, and environment, it becomes possible to formulate a *model* to facilitate analysis.

2.1.2 ABO Modeling and Analysis. In their discussion of ABO modeling and analysis, Rich et al. emphasize management's requirement for assessing the extent to which an aircraft support system can meet operational requirements (1987:3). In taking the system postulated above as given, the issues of force structure and force modernization may be excluded to permit analysis to directly focus upon aircraft support system assessment. The issue thus becomes the discovery of those features most critical to mission support for a given system and resource set, as measured in terms of system response. In this research, the interpretation is clear--the air base, as a system, responds to resource level changes in terms of changes in the number of sorties flown within a given time interval. Given that the environmental variables of attrition and attack or no-attack scenario are considered uncontrollable factors, the logical candidate for system experimentation is resource level variation (manipulation)--the analysis of resource policy alternatives.

Evaluating the impact of resource policy alternatives on logistics system output requires careful modeling and analysis in lieu of field experimentation, due to the potential severity of interruptions to air base operations. Several researchers have proposed numerous models and analysis approaches for the task; Diener notes that the

TSAR/TSARINA simulation model set was specifically designed for such research:

> TSAR was created with the interdependencies of air base resources as a focal point. The intent was to permit decisionmakers to explore the air base as a system in order to seek improvements to that system. (1989:3)

With TSAR (simulating sortie generations) and TSARINA (simulating enemy attacks) providing samples for analysis, it becomes possible to identify and evaluate those resources most critical to mission support at different points in time. Specifically, air base behavior may be analyzed, at a high level of detail, in terms of a treatment/response model, where the analysis of sortie generation (response) variance occurring throughout a specified period of time is essentially enabled *via* imposing different initial conditions (treatments) in resource levels (factors). The ultimate aim of this formal experiment lies in providing managers with tools and results that will facilitate informed decision making:

> Management within the services' logistics functions actions ought to be based (to the extent possible) on good estimates of their [policy alternative] effects on overall system outputs-available aircraft and sortie generation capability in wartime. (Rich et al, 1987:7)

The following discussion of Diener's research approach (1989), a controlled statistical experiment, is intended to provide an understanding of the sample generation and analysis processes from which this study takes its genesis. Of particular importance are the resources used as input factors to the model, the experimental design, the issue of variance, and the sample organization used in the subsequent development of simulation metamodels.

2.2 Previous Study Statistical Experiment

2.2.1 Factors.

2.2.1.1 Resource Representation. A complete description of each experimental resource factor, low and high levels, as well as the rationale used in determining their respective assumed values or interpretation (in terms of resource management policy) can be found in Diener (1989:27-42). From a modeling perspective the factors are, by definition, categorical; the actual metric value changes occurring throughout the 30 day simulation horizon are largely transparent to the researcher. This representation schema is congruent with a high-level resource management policy perspective--each treatment (run) constitutes an initial air base management policy whose effects (or consequences) are evaluated in terms of the number of sorties generated by the base for each day of the one month period. Due to their inherently non-metric characteristics, factors are thus represented as indicator variables in both the simulation used to generate analysis samples and in the regression metamodels subsequently used for describing the relationship between resources and sorties flown.

The resources used in sortie generation are highly interactive within the dynamic environment of an air base. To render the analysis of their temporal behavior tractable, the exploratory research performed by Diener is scoped to consider the importance of factors taken singly (main effects) and two at a time (two-way interactions). Further, what the factors represent is indicative of the level of detail the model intends to capture: "Each factor also represents an actual functional area found within the logistics infrastructure of a tactical air base" (Diener, 1989:29). The simulation attempts to explore the behavior (in terms of main effects and two-way interactions) of entire *functional areas*, and does not attempt to model or predict the specific behavior of a particular functional area in the absence of others. The entity in question is thus the entire logistics infrastructure that supports the sortie generation process:

> To be realistic and useful, the derived metamodel must capture a wide spectrum of various logistics resource positions as well as elements of an uncertain wartime environment such as air base attack and attrition. Thus the high and low levels for each variable are chosen so that we have a valid and realistic inference space. (Diener 1989:29)

2.2.1.2 Factor Levels. The specific resource factor definitions used in Diener's experiment are based on a high and low level for each resource and environmental factor (see Tables 2.1 and 2.2). In general, high level factor settings "represent the logistics infrastructure one would expect to find supporting 72 F-15 aircraft", and "given the 'high', it seems logical to then degrade it to develop the 'low' level for each resource", (Diener, 1989:29). While the reader is referred to Diener (1989) for more detailed descriptions of the factors listed in Tables 2.1 and 2.2, the following characteristics differentiating initial resource level settings (prior to simulation) are noted:

The environmental factor of aircraft attrition (factor A) "...is modeled as a stepwise reduction of the attrition rate from 1.2% to 1.0% of sorties flown", whose timing is based on the rationale that "...the more we fly, the less effective the enemy is against us, and thus the attrition rate is driven down", (Diener, 1989:41). In the low case, attrition remains at 1.2%.

Available Aircraft (factor B) is initially distinguished by the presence or absence of filler aircraft, while the levels of Aircraft Battle Damage Repair (ABDR) Capability (factor C), Personnel (factor E), Avionics Intermediate Test Stations (AIS) (factor F), Support Equipment (factor G), and Spares (factor H) are differentiated in terms of a percentage or arithmetic reduction in number.

In regard to Recovery (repair of damaged runways and taxiways) (factor D), Missiles (factor J) and Fuel (factor K), level differentiation is accomplished temporally: recovery procedures are slower at the low level (due to the use of alternate procedures), and quantities of Fuel and Missiles are diminished at low levels *via* less frequent and smaller deliveries.

Table 2.1

Resource Factors - High Levels (Diener, 1989:30)

Factor	Description
B - Aircraft	72 assigned plus 18 filler aircraft available with 72 hour delay
C - ABDR Capability	6 assessors (2 per AMU) where work cannot begin until damage is inspected by a trained assessor
D - Recovery	Full range of improved procedures which includes manual workarounds; CE and EOD personnel equipment also included
E - Personnel	Typical quantities expected -- assume these are the number authorized by specialty
F - AIS	2 sets with 5 stations each
G - Support Equipment	Typical quantities expected -- assume these are the number authorized
H - Spares	Computed by TSAR with 100% safety factor using AFM 67-1 Chapter 11 procedures
J - Missles	Initial Stocks: 300 AIM9-M 300 AIM7-M Initial Components: 612 AIM9-M 424 AIM7-M Deliveries: Day 1 -- AIM9-Ms and AIM7-Ms Days 2,5,10,15 -- AIM9-M and AIM7-M components
K - Fuel	Deliveries arrive Days 10,15,20, and 25

Table 2.2

Resource Factors - Low Levels (Diener, 1989:31)

Factor	Description
B - Aircraft	72 assigned, no filler aircraft
C - ABDR Capability	3 assessors (1 per AMU) where work cannot begin until damage is inspected by a trained assessor
D - Recovery	Slower alternate procedures; CE and EOD personnel and equipment reduced to 75% of high level
E - Personnel	Quantities reduced to 75% of high level
F - AIS	1 set with 5 stations each
G - Support Equipment	Quantities reduced to 75% of high level
H - Spares	Computed by TSAR with 10% safety factor using AFM 67-1 Chapter 11 procedures
J - Missles	*Initial Stocks:* same as high case Deliveries: Days 5,10,15 -- AIM9-M and AIM7-M components
K - Fuel	Deliveries arrive Days 10 and 20

Finally, it should be noted that resource factor levels are subject to impacts from (conventional) attacks. Diener notes that "Six attacks occur in the first five days..." and that "The attacks are optimized from the enemy's perspective with regard to aimpoints and time of attack", (1989:39). His tabular summary of the conventional attacks occurring within first 5 days of the Attack Scenario are listed in Table 2.3.

Table 2.3

TSARINA Simulated Attack Summary (Diener, 1989:40)

Day	Hour	Attackers	Munitions	Targets
1	0550	10 bombers	24 bombs each	R,T
		5 bombers	24 mines each	R,T
	1450	8 fighter-bombers	10 bombs each	R,T
		4 fighter-bombers	10 mines each	R,T
		24 fighter-bombers	1 bomb each	A
2	0550	8 fighter-bombers	10 bombs each	R,T
		4 fighter-bombers	10 mines each	R,T
		24 fighter-bombers	1 bomb each	A
3	0550	4 bombers	24 bombs each	R,T
		1 bomber	24 mines	R,T
		10 bombers	24 bombs each	S,A
4	0550	4 bombers	24 bombs each	R,T
		1 bomber	24 mines	R,T
		10 bombers	24 bombs each	S,A
5	0550	4 fighter-bombers	10 bombs each	R,T
		4 fighter-bombers	10 bombs each	S,A

Legend for Targets:

Symbol	Description
R	Runways
T	Taxiways
A	Aircraft
S	Support Fa-cilities

2.2.2 Experimental Design. As noted, the TSAR/TSARINA large-scale simulation experiment documented in Diener's work provided the data sample from which regression meta-model beta coefficients were estimated. The sample was generated *via* a fractional factorial design for factors at two levels (Diener, 1989:44; McLean and Anderson, 1984).

Regression beta coefficient estimates were derived and subsequently used for analyzing resource factor contributions to sortie generation for each day of the analysis horizon, and for predicting sorties flown (Diener, 1989:110-216).

It must be clearly understood that while regression was utilized for analysis, a fundamental aim of Diener's research is the explanation of response variance, where the experimental unit is the F-15 air base, the factors are identified as the 10 independent variables previously discussed, and the response is the dependent variable measuring sorties flown. The analysis of variance (ANOVA) technique is applied *via* the use of multiple linear regression (note that ANOVA models can be shown to be mathematically equivalent to a corresponding regression model). In this way, a single model (for each day of the 30 day simulation horizon) serves to explain response variable variance *and* predict values of same.

With respect to sampling plan structure, the specific experimental design and factor treatment combinations sampled can be found in McLean and Anderson, Plan 8.10.16 (1984:265-266). Using low and high levels for each of ten factors (nine resource and one environmental), a Resolution V design was employed to ensure that confounding of main effects and two-way interaction terms did not occur within the data sample generated for the subsequent analysis of variance *via* regression (Diener, 1989:42-44).

Each treatment factor combination (simulation design point) essentially defines an input vector specifying the initial conditions (resource levels) for a single run; 128 treatments (runs) were required for each scenario (attack and no-attack), for a total of 256 treatments (Diener, 1989:44; McLean and Anderson, 1984:256). The fractional factorial plan employed is a 1/8 replication of 10 factors in 8 blocks of 16 units each (McLean and Anderson, 1984:256) (blocking is discussed below due to its relevance to simulation variance reduction). A full factorial design for this experiment would require 2^{10} simulation runs for each scenario (1024 possible treatments defined by all possible combinations of 10 factors at two levels for each scenario, 2048 total), thus the 128 treatments per scenario used by Diener constitutes a 1/8 fraction of the full factorial design. The actual identification of *which* treatments are included in the sample is a function of fractional factorial design construction process; the reader is directed to Montgomery (1991:514-528) and McLean and Anderson (1984) for specific examples.

The Resolution V design employed isolates, i.e., allows one to estimate, the contribution (effect) of each main effect and two-way interaction: with respect to a particular fraction's (sample) composition, the orthogonal properties of the selected experimental design ensure that each factor is given "equal" representation throughout the sample with respect to all other factors. With respect to a particular sample's representativeness in relation to the underlying population probability distribution, the issue becomes one of assessing the extent to which a particular fractional (being a member of the set of all possible fractional samples) is in fact representative. In the interest of economy, a single fractional factorial design is often employed to ensure sufficiency of resolution for estimating main effects and two-way interactions, with full recognition that the particular sampling plan selected is only one of a large set of such plans. In this way, the potentially prohibitive cost of running a full factorial simulation is avoided, while the representativeness of the sample is assumed sufficient for the preliminary task of identifying and/or screening significant factors (cost is a fundamental consideration, especially for full factorial designs and/or designs with several factors/factor levels). Thus, the selected design provides an economical means for obtaining a representative sample for the factors of interest. With this assurance (and caveat), the next issue for consideration entails controlling the degree of model variance introduced on the response variable sorties flown. Accordingly, the focus turns to a discussion of variance and how model variance may be controlled during the simulation experiment.

2.2.3 Variance. In their discussion of variance reduction techniques, Law and Kelton note:

> One of the points we have tried to emphasize throughout this book is that simulations driven by random inputs will produce random output. Thus, proper statistical techniques applied to simulation output data are imperative if the results are to be properly analyzed, interpreted, and used (see Chaps. 8, 9 and 12). ...If we can somehow reduce the variance of an output random variable of interest...without disturbing its expectation, we can obtain greater precision, e.g., smaller confidence intervals, for the same amount of simulating or, alternatively, achieve a prespecified precision with less simulating. (1991:612-613)

In regard to the distinction between true and experimental error variance, Diener notes that problems in sortie generation modeling generally stem from variability due to the nature of the data and processes to be modeled, and from variability due to the model itself. Variability due to the (simulation) model can stem from non-constant variance both between and within scenarios: (1) testing response variable estimates (sorties) for conditions of normality may reveal that they are not identically and independently distributed (IID) across design points (recall that each estimate has its own probability distribution), (2) common random number streams used across all design points can induce correlation, and (3) time series response autocorrelations are present within a run (treatment) (Diener, 1989:56-59).

2.2.3.1 Variance Reduction. With respect to controlling variance *via* variance reduction techniques

(VRTs), Diener states: "By blocking based on random numbers, we can usually reduce the experimental error as compared to a completely randomized design" (1989:46). The implementation of the above sampling plan and a VRT are accomplished as follows: each scenario set of 128 treatments is divided into subsets of 16 units, where each 16-unit treatment subset is assigned one of eight different random number seeds in TSAR (Diener, 1989:22-26,44-49); the specific treatments comprising each block are in accordance with the design characteristics of Plan 8.10.16 in McLean and Anderson (1984:265-266). Treatments are thus assigned in groups to each stream. The VRT is introduced through the use of common random numbers (CRN) across all runs (treatments) within a block, on the theoretical basis of a common starting seed. Thus correlation is induced within each of the eight blocks, with the theoretical result of reduced variance of the statistical estimator of sorties flown. A similar technique is used for the attack scenario, where both TSAR *and* TSARINA are employed:

> In the attack case, each block has the same attack in terms of targets and number of attacking aircraft, but each has a different randomly selected starting random number seed within the TSARINA model. This gives us eight different random versions of the same attack. (Diener, 1989:48)

The principal idea in the attack case is that within each block, both TSAR and TSARINA use a CRN technique to theoretically reduce the variance of the statistical estimate of the response (sorties flown), at least in terms of a common starting seed shared among the runs (treatments) contained within the block. In the interest of clarity, it should be noted that TSAR and TSARINA, when jointly employed, do not share the same random number stream--each simulation model component uses its own dedicated stream, and, as in the attack scenario, is subject to possible occurrence of within-block divergence discussed immediately below.

It is noted that this CRN technique may not always be effective for TSAR/TSARINA due to the nature of the model:

> In the no-attack case, each block is defined by a different starting seed (randomly selected) for the TSAR random number stream. The random numbers for all random events are drawn from this stream...The flow of random numbers is not congruent from run to run because malfunctions cause additional random numbers to be used...[In the attack case]...The use of random numbers would be congruent case to case except that a probability of arrival is also randomly checked for each attacker. (Diener, 1989:47-49)

Noting these potential difficulties in applying the CRN, a review of its effectiveness is in order.

2.2.3.2 <u>Efficiency of Variance Reduction</u>. The fractional factorial experimental design and the CRN VRT approach based on blocking and common random number seeds were jointly employed for realizing a major research objective of Diener's work: "A major objective of this research is to apply an experimental design that reduces the number of runs as much as possible while achieving an acceptable level of experimental error" (1989:52). In his evaluation of the overall success of the variance reduction and design efficiency of same, it is observed that

> The success of the variance reduction technique in this [no-attack] case ranges from a high of 43.72% for the Day 1 results to -4.91% for reduction Day 22 [sic]. The average variance reduction across all thirty models is only 5.68%...The success of the variance reduction technique is more apparent in this [attack] case, ranging from a low of 18.92% on Day 8 to a high of 81.90% on Day 2. The average percent reduction in variance is 38.81%. (1989:68,72)

Thus, in general, the variance reduction obtained by the CRN approach is significant for the attack scenario. Its success over the no-attack scenario is explained in terms of blocking both over the random number streams in TSAR as well as the steams in TSARINA, in terms of a common random number starting seed.

2.2.4 <u>Correlated Responses and Sample Organization</u>. As noted, the statistical experiment produces a total of 256 samples (1 per treatment, 128 per scenario). A single treatment specifies the initial resource levels (low/high combinations) for a run, and the simulation subsequently generates a time series of 30 responses (sorties flown measures), one response for each day of the one month simulation run span. The response vector variance is attributed to true and experimental error, as well as potential time series autocorrelations.

If organized by day *cross-sectionally*, a daily sample contains 128 treatment/response observations, where each observation consists of one treatment vector and an associated number of sorties flown. Using this method of organization, there are a total of sixty daily sample sets, (30 for the attack scenario days, 30 for the no-attack days). This cross-sectional sample organization was used by Diener to develop regression metamodels for each day of each scenario (60 metamodels in total). This method of sample organization is useful in dealing with problems stemming from autocorrelations when conducting time series analysis: by analyzing treatment/response combinations by *day* cross-sectionally, the requirement for applying time series analysis techniques to remedy autocorrelation is mitigated to some extent, as the time series reduces to a single response per treatment using cross-sectional sample organization. This does not suggest that autocorrelation is not present in the longitudinal sense but rather that the analysis approach becomes one of applying regression techniques to the sample of observations created for each day when cross-sectional organization is applied.

Diener suggests that a multivariate stochastic model (which assumes feedback between inputs and outputs) or an intervention model would be useful in future research efforts, for attempting to model the air base system behavior in question in terms of a time series (1989:234-239). Both in theory and practice, the difficulty and newness of such methods must be noted. In his prognosis of transfer function methodology, Makridakis et al. state:

In order to appreciate transfer function modeling (or MARIMA [Multiple Autoregressive Integrated Moving Average modeling]), a considerable amount of careful study is required... Multivariate extensions are being studied in academic settings and it is expected that more significant use will be made of the full richness of the TF [Transfer Function] model in the near future. (1983:534)

Diener's experiment focuses on the identification of main effects and significant two-way interactions, an analysis task best characterized as exploratory. Given this task and its relative magnitude, it is important to recognize that the preliminary identification of significant factors must be performed prior to the application of such theoretically based approaches to analysis (as MARIMA and TF methodologies), if the research problem is to be tractable. Accordingly, the analysis approach used by Diener focuses on the identification of significant factors. Figures 2.1 and 2.2 represents the entire sample set generated by the simulation for both scenarios, in terms of a three dimensional representation, for sampled treatments only (assigned case numbers 1 through 128).

As illustrated, the individual treatment time series do exhibit very strong autocorrelations in that the general trend for both scenarios is one of a generally smooth decline in sorties flown throughout the 30 day simulation horizon. In addition, the Attack Scenario responses appear to exhibit more variation than their No-Attack counterparts. Accordingly, Diener's decisions to model the two scenarios separately as well as cross-sectionally are appropriate.

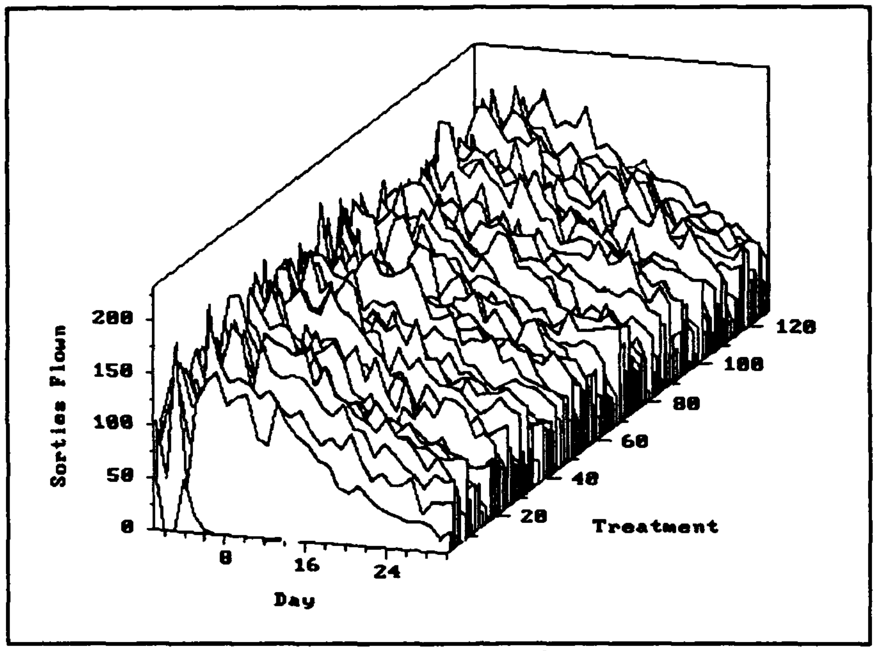


Figure 2.1. Attack Scenario Sorties Flown Time Series

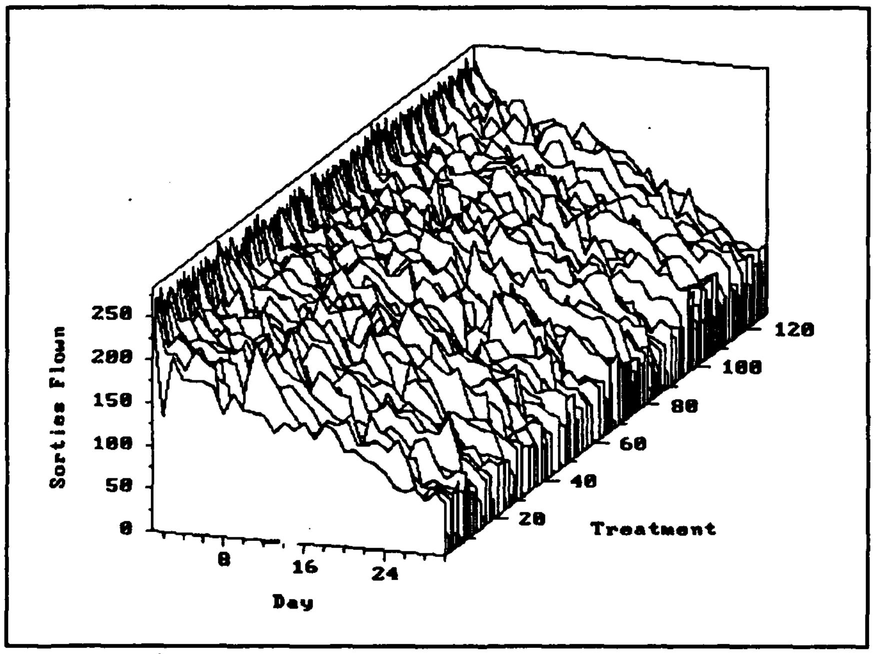


Figure 2.2. No-Attack Scenario Sorties Flown Time Series

2.3 Metamodels

A metamodel can be understood as a simpler model of a model. In Diener's words:

> Although the simulation model is a simpler representation of reality, it can be very complex in and of itself. Thus an even simpler model may be used to better understand the complex model; this simpler, auxiliary model is often called a metamodel. (1989:6)

Friedman states that metamodels are extremely useful for understanding and exploring a more complex model, citing several researchers that support the contention (Friedman, 1983:28-31). Further, as discussed in Friedman and subsequently adopted by Diener, there is a direct relationship between the modeling forms for a real system, simulation model and an analytic metamodel (Friedman, 1983:43; Diener, 1989:7). Prior to discussing this relationship, the importance of metamodels for applied research is noted.

2.3.1 Metamodels and Applied Research.

The practical significance of metamodeling is at least two-fold: (1) it provides an alternative to the repeated application of time and cost-intensive large-scale simulations, and (2) it permits the researcher to focus on specific response *regions* of the system under study. In the first case, analytical metamodels provide a useful research tool in cost-constrained environments; they can be critical in real-time applications where issues of survival render the alternative of time-intensive simulation untenable. In the second case, for example, an entire field of research known as response surface methodology (RSM) fits a model to data (most often simulated) in efforts to determine the optimum levels of input factors with respect to a response variable. In this and similar analytic modeling techniques, specific regions of interest become visible as the result of their analytical representation.

It should be noted that many analytic metamodels are derived *via* statistical analysis and fitting of samples provided by simulation. The point here is not to replace simulation, but rather to recognize the benefits of applying analytic techniques in environments that are time and cost sensitive and where the use of supplemental analytical tools is advantageous. Morgan and Henrion note the following in their discussion of techniques and tools available for uncertainty analysis, which illustrates the benefits of employing a combination of approaches:

> Combinatorial scenarios [in simulation] will cover a larger part of the model behavior, but require far too many model runs in general. Fractional factorial designs (FFD) (Box, Hunter and Hunter, 1978) have been quite popular. These select a subset of the combinatorial scenarios...The main purpose of the sampling process is to identify those uncertain inputs that contribute most to the output, that is, essentially perform uncertainty analysis. Usually just a few uncertain inputs are found to contribute the majority of uncertainty in the output. The simplified response surface need model only the effect of these, and can generally ignore the other inputs. (1990:210-211)

2.3.2 <u>The Modeling Hierarchy</u>. The *real system* (physical reality) form, a relationship between an unknown number of factors and a system response, is postulated as

$$\tau = g(x_1, x_2, \ldots x_q) \qquad (2.1)$$

where

τ	= the physical system response,
$x_{1 \text{ to } q}$	= the q environmental or controllable input factors, and
$g()$	= the unknown relationship between the response and the q factors, where the value of q (the number of factors) is unknown. (Friedman, 1983:26)

A *simulation model* can be conceptualized as an approximate functional representation (as a response function) of the true system formalized in Eq (2.1) (Diener, 1989:23). As a primary, or first-order, model, it estimates the real system by approximating g() in terms of a postulated function f(), where f() represents the entire simulation, including all stochastic processes and random variate interactions. The number of q (input) factors is set at a specific constant, say k. Thus, the simulation model estimate of the real system can be represented as

$$y_i = f(x_{i1}, x_{i2}, \ldots x_{ij}) + \epsilon_i \qquad \begin{array}{l} i = 1, \ldots n \\ j = 1, \ldots k \end{array} \qquad (2.2)$$

where

y_i	= the simulation response in the ith replication,
$x_{i,j}$	= the value of the jth factor in the ith replication, and
ϵ_i	= the error term in the ith replication (Friedman, 1983:30)

Finally, an analytic *metamodel* of the simulation process given in Eq (2.2) can be understood as an attempt to simplify the stochastic behavior of the simulation in terms of an analytic form (note the need for replication with respect to parameter estimates from which to derive the metamodel). In this sense the simulation model is a primary or first-order model, while the analytic model is a second-order model (hence the concept *metamodel*). Friedman notes that several researchers favor the additive model of experimental design (the general linear metamodel) in initially selecting a metamodel for exploratory system representation (1983:29); this model form was adopted and employed by Diener (1989). The initial metamodel form used in such research is given by Friedman as

$$y_i = \beta_0 + \Sigma(\beta_j x_{ij}) + \epsilon_i \qquad \begin{array}{l} i = 1,\ldots,n \\ j = 1,\ldots,k \end{array} \qquad (2.3)$$

where

y_i = the response value of the ith observation,

β_0 = a weight constant (parameter) understood as the y-intercept when all other x_js = 0,

β_j = the weight constant (parameter) for the jth factor x_j,

x_{ij} = the value of the jth factor in the ith observation, and

ϵ_i = the unknown error term for the ith observation. (1983:29-30)

Friedman postulates a clear relationship between the design of simulation experiments, the application of a metamodel, and the application of statistical tests to the experimental results:

> Depending on the experimental layout, whether the factors are quantitative or qualitative, and the aim of the study, the general linear metamodel (linear in the parameters, not necessarily in the x's)...may be applied to regression analysis, analysis of variance, analysis of covariance, t-test, paired t-test, etc. Whether a researcher explicitly says so or not, designing simulation experiments which will be analyzed via one of these statistical tests implies the use of this general linear metamodel in one of its forms... (1983:31)

A important point for this research lies in this observation: in general terms, the analysis of data *via* statistical techniques presupposes the existence of a *specific* relationship between variables. Most often, this relationship is represented in terms of an *assumed* functional form. Thus, prior to analysis, a functional relationship is assumed to exist, *a priori*, among variables. This in turn suggests that the characteristics of a selected experimental design are largely defined by the assumed functional form of the relationship of variables under study. The key point is that a functional form is assumed prior to sampling, modeling and analysis activities, *de facto*. Thus, a brief examination of regression metamodel forms, in relation to experimental design considerations, is in order.

2.3.3 Metamodels and Experimental Design. In their discussion of factorial designs, Law and Kelton note the following regarding a factorial design for an (s,S) inventory model for two factors at two levels each:

Calculation of main effects and interactions of 2^k factorial experiments is actually equivalent to estimating the parameters in a particular statistical regression model of how the response depends on the factors. For Example 12.1 this regression model is

$$R(s,S) = a + bx_s + cx_S + dx_s x_S + \epsilon$$

where R(s,S) = response as a function of s and S... (1982:376)

Note that this two factor level regression model contains main effect and secondary interactions terms--the terms of interest to Diener. Here there is a direct correspondence between the design of the factorial experiment and the *general form* of the regression metamodel selected, as implied by Friedman (1983:31). Further, this correspondence demonstrates the congruence of Diener's selection of model form for exploratory analysis (of main effects and secondary interactions) with the observations of McLean and Anderson (1984) previously noted. Finally, the assumption of a specific functional form prior to *sampling* is clear.

Although expanded to include additional factors (10 total), Diener's metamodels possess essentially the same form as Law and Kelton's above, and include terms for main effects, secondary interactions, error, and an additional variable added to account for the effects of blocking. The no-attack and attack scenario metamodel forms are given in Diener (1989:49-51) as Eq (3.3) and (3.4), and are duplicated here as Eq (2.4) and (2.5). The metamodel form for the no-attack scenario is given as

$$S_i = \beta_0(i) + \sum_{j=1}^{10} \beta_j(i)x_{1j} + \sum_{j=1}^{9} \sum_{k=j+1}^{10} \beta_{jk}(i)x_{1j}x_{1k} + B_{11}(i) + \epsilon(i)$$

$$i = 1,\ldots,30 \quad (2.4)$$

where

S_i = the number of Sorties flown on Day i,

$\beta_0(i)$ = the y-intercept when $\forall(j>0)(\beta_j = 0)$,

$\beta_j(i)$ = the degree of importance of variable x_{1j} on the variation of S_i,

$\beta_{jk}(i)$ = the degree of importance of the two-way interaction of variables x_{1j} and x_{1k} on the variation of S_i,

x_{1j} = the level (indicator variable) of factor x_j on Day 1,

$B_{11}(i)$ = reflects the random effect of blocking on Day i due to the random number streams in TSAR, where

$$B_{11}(i) \sim N(0,\sigma^2_B), \text{ and}$$

$\epsilon(i)$ = reflects the experimental error, where

$$\epsilon(i) \sim N(0,\sigma^2_\epsilon)$$

The metamodel form for the attack scenario is given as

$$S^*_i = \beta^*_0(i) + \sum_{j=1}^{10} \beta^*_j(i)x_{1j} + \sum_{j=1}^{9} \sum_{k=j+1}^{10} \beta^*_{jk}(i)x_{1j}x_{1k} + B^*_{11}(i) + \epsilon^*(i)$$

$$i = 1,\ldots,30 \quad (2.5)$$

where

S^*_i = the number of Sorties flown on Day i,

$\beta^*_0(i)$ = the y-intercept when $\forall(j>0)(\beta^*_j = 0)$,

$\beta^*_j(i)$ = the degree of importance of variable x_{1j} on the variation of S^*_i,

$\beta^*_{jk}(i)$ = the degree of importance of the two-way interaction of variables x_{1j} and x_{1k} on the variation of S^*_i,

x_{1j} = the level (indicator variable) of factor x_j on Day 1,

$B^*_{11}(i)$ = reflects the random effect of blocking on Day i due to the random number streams in TSAR and TSARINA, where

$B^*_{11}(i) \sim N(0,\sigma^2_{B^*})$, and

$\epsilon^*(i)$ = reflects the experimental error, where

$\epsilon^*(i) \sim N(0,\sigma^2_{\epsilon^*})$

The metamodel forms in Eq (2.4) and (2.5) can be viewed as *class definitions*: they serve as forms to guide the researcher (at least implicitly) in the design of the experiment, and define the set from which specific models are selected (the selected models may be understood as instances of the defined class).

In Diener's metamodels, the no-attack scenario metamodel *form* given in Eq (2.4) is comprised of the representation of the main effects of 9 resources and the environmental variable of aircraft attrition, and all possible two-way interactions of the 10 main effects. The attack metamodel *form* given in Eq (2.5) is identical to that of the no-attack metamodel, with the caveats that the response variable, all beta coefficients, the blocking variable, and the error term are all modified by the superscript "*", denoting the additional influence of attacks (induced by TSARINA) on parameter estimates. Thus, both attack and no-attack metamodels are identical in form, but are different in terms of parameter value estimates.

2.3.4 Functional Foundation of Modeling and Analysis. In light of the previous presentation of the metamodeling hierarchy and the specific analytic metamodels forms selected by Diener, it becomes possible to logically deduce and explicitly state a key point regarding the modeling process: when modeling a physical system, the functional form explicitly or implicitly selected by the researcher determines, in general, the sampling plan, experimental design, and statistical techniques that will be used to analyze that system, i.e., the viability of the modeling process is contingent upon the functional form selected prior to modeling and analysis. In Diener's research, the models are assumed to be linear, with significant two-way interaction terms; the deduction above is applicable to his research by virtue of the mathematical equivalence between ANOVA and regression models. Given this selection of form, the remaining issue becomes the selection of specific metamodels for use in analysis. In the following discussion regarding the metamodel selection process, the focus is strictly limited to those models that are linear in the betas (the β parameters), as this characteristic is assumed in Eqs (2.4) and (2.5).

2.3.5 Metamodel Selection. In regard to the selection of a particular model from the class of metamodels defined by a regression metamodel *form*, several selection rules are possible, including techniques such as backward, forward and/or stepwise elimination of variables, highest R^2, and others. In Diener's research, the specific daily regression metamodels are selected *via* backward elimination of variables. It is usually the case that one cannot determine the *best* model, but rather chooses the best from a set of candidates (Meyers, 1986:100). Two possible reasons for this difficulty are (1) the class definition of the model form selected precludes other model forms from consideration (for example, only linear forms are assumed), and (2) the number of possibilities generated from a selected form is too large to enumerate and subsequently analyze in terms of a decision (selection) rule.

Given that the metamodel form has been specified as linear, we focus next on metamodel assessment measures, to illustrate that metamodel selection is contingent upon analysis requirements. In particular, the point is made that goodness-of-fit is not equivalent to goodness-of-prediction, by way of the comparison of two statistical measures--one used for fitting, the other for assessing predictive potential.

2.3.6 Assessing Fit and Predictive Potential. Clearly, metamodel predictive capability can be assessed on the basis of empirical performance (*a posteriori*) through testing and validation procedures involving split samples, 10-fold validation, and other techniques (Meyers, 1986; Weiss and Kulikoski, 1991). The question is whether predictive capability can be assessed during the model fitting phase before testing or validating the model on independent samples, i.e., whether metamodel predictive potential can be assessed *a priori*.

With regard to standard practice in regression model *fitting*, one of the most often used statistics is the coefficient of determination, or R^2 (or equivalently, the square of the coefficient of correlation). This statistic rests upon fundamental concept of *partitioning total variability* in terms of the regression model. Expressed as a functional relationship, it is given by Meyers (1986:16) as

$$\sum_i (y_i - \bar{y})^2 = \sum_i (\hat{y_i} - \bar{y})^2 + \sum_i (y_i - \hat{y}_i)^2 \qquad i = 1,\ldots,n \quad (2.6)$$

or, equivalently, as $SS_{Total} = SS_{Reg} + SS_{Res}$

where

SS_{Total} = the total variability observed,

SS_{Reg} = the variability explained by the model, and

SS_{Res} = the variability not explained by the model.

R^2 can then be formulated (Meyers, 1986:28-31) as

$$R^2 = SS_{Reg} / SS_{Total} = \sum_i (\hat{y}_i - \bar{y})^2 / \sum_i (y_i - \bar{y})^2 \qquad i = 1,\ldots,n \quad (2.7)$$

or, equivalently, as $R^2 = 1 - (S_{Res} / SS_{Total})$

where

SS_{Res} = the variation unexplained, and

SS_{Total} = the total variation observed.

R^2 is thus the proportion of variability explained or accounted for by the model. It is important to note that *all* available data points are used in the statistic's derivation--all values are used for *fitting*.

In regard to assessing metamodel a *priori* predictive capability, a set of residuals is required such that the observation(s) used in fitting *are not the same* as those used for assessing predictive potential. The Prediction Sum of Squares (PRESS) statistic, R_{Pred}, which is based on a "leaving-one-out" validation procedure, offers a potential solution: An observation y_i is omitted, the model is fit (the parameters are estimated) with the remaining n-1 data points, and a prediction for the omitted observation is generated. In this way, n PRESS residuals are obtained and a PRESS Sum of Squares may be derived; Meyers lists it as

$$PRESS = \sum_i (y_i - \hat{y}_{i,-i})^2 \qquad i = 1,\ldots,n \quad (2.8)$$

where

$\hat{y}_{i,-i}$ = the prediction generated by the candidate model whose parameter estimates are derived in the absence of y_i.
(106:87)

The PRESS Sum of Squares provides the means for the calculation of a PRESS statistic in a manner analogous to the computation of the correlation of determination, or R^2. The statistic R_{Pred} is thus given (Meyers, 1986:107) as

$$R_{Pred} = 1 - \text{PRESS} / \sum_i (y_i - \bar{y}_i)^2 \qquad i = 1,\ldots \quad (2.9)$$

Note that the estimate for a given response is derived in the absence of that response. Clearly, the candidate model with the smallest R_{Pred} should be considered as a potentially good predictor, although many other factors must be considered (Meyers, 1986:111-133). Finally, when sufficient data are available, data splitting validation techniques become viable (using an independent test data set to validate a fitted model's predictive performance), which can provide the analyst a clear and direct assessment of a candidate model's predictive capability.

The coefficient of determination (R^2) and the PRESS statistic (R_{Pred}) are presented here (Eq(2.7) and (2.9), respectively) to emphasize that at least three logically distinct possibilities (goals) exist for the researcher when modeling a system in terms of a functional relationship: the researcher may wish to *explain* (fit) a system, predict the future behavior of that system (assuming a causal relationship exists), or, in the interest of parsimony, may wish to do *both* with the same model. In Diener (1989), the emphasis is clearly on explanation, although prediction is listed as a research issue (1989:9). A seminal issue for this research is simply whether or not a specific non-linear modeling technique will yield better results in predicting and explaining the air base system behavior than Diener's metamodels. Accordingly, the focus of discussion turns to the conceptual background that underlies this method, which is commonly referred to as artificial neural network modeling.

2.4 Artificial Neural Network Concepts

Artificial neural systems, also referred to as neural networks, are a recent and important development. Neural networks stand apart from other artificial intelligence techniques in their ability to detect patterns hidden in many forms of data. Fundamentally, a neural network is a pattern classification system that maps input to output patterns. Generalizing this concept for any number of inputs and M output classifications, Lippmann states:

> The goal of pattern recognition is to assign input patterns to one of a finite number, M, of classes ... Input patterns can be viewed as points in multidimensional space defined by the input feature measurements. The purpose of a pattern classifier is to partition this multidimensional space into decision

regions that indicate to which class any input belongs. (1989:47-48)

2.4.1 Modeling the Brain. Artificial neural systems researchers attempt to model human learning processes by constructing simplified models of the human brain (Hecht-Nielsen, 1988:37). "This idea of comparing the central nervous system to the currently most complex information processing systems currently available can be followed throughout the literature" (Rogers et al., 1990:6). In spite of the dangers of over-simplification (Schwartz, 1988), a model of a biological system often serves as a springboard for research (Baum, 1988:205). Cowan and Sharp credit the birth of symbolic neuronal modeling to McCulloch and Pitts:

> Perhaps the first major contribution was in a paper by Warren McCulloch and Walter H. Pitts published in 1943. In this paper McCulloch and Pitts applied symbolic logic to the problem of describing what neural nets can do. In effect they proved that all processes that can be described with a finite number of symbolic expressions ... can be embodied in nets of what they call "formal" neurons. (1988:86)

From this, subsequent research efforts focused on representing neuronal processes mathematically, using formal neural network representations for the research framework:

> McCullogh and Pitts left a strong imprint on the science of neural nets with their seminal study in 1943 of the mathematical representation of neuronal modelling. It was not their model *per se* that led others to attempt mathematical neural net descriptions, but the evidence that such descriptions could be fruitful. (Barron et al., 1987:1)

2.4.2 Learning and Representation.

Learning is clearly critical for survival in that it provides a means by which organisms adapt to the environment. One of the first formulations of a biological learning law was offered by Hebb: "In 1949, Hebb proposed that the connectivity of the brain is continually changing as an organism learns differing functional tasks and that cell assemblies are created by such changes" (Cowan et al., 1988:88). In terms of a process, the Hebbian learning law states that "...when a cell A repeatedly participates in firing cell B, then A's efficiency in firing B is increased" (Khanna, 1990:11). This increase in the firing rate (efficiency) of cell A can be represented by increasing the value of the weight of the synapse connecting neurons A and B (Lippmann, 1987:13-14). While many new learning paradigms have been developed (see Hecht-Nielsen, 1990:46-77, for a broad review) that are, in general, much more powerful than the original Hebbian law, the fundamental concept of network learning can be understood in terms of synaptic weight adjustments: "The ability of the networks to *learn* actually refers to the networks adapting their internal adjustments or weights following exposure to the data" (Rogers et al., 1990:2).

The development of an artificial neuron called the *perceptron* by Rosenblatt in 1959 was a milestone in neural network research (Rogers et al., 1990:13,41). Expanding the domain investigated by McCullogh and Pitts, Hebb, and others, Rosenblatt explored and developed several neural network architectures and learning rule properties that collectively defined a paradigm for neural network research (Pao, 1981: 115-123; Pineda, 1989:50). Through his and other research efforts, a distinction between representation and learning became apparent:

> The proof of the perceptron learning theorem (Rosenblatt 1962) demonstrated that a perceptron could learn anything that it could represent. It is important to distinguish between representation and learning. Representation refers to the ability of a perceptron (or other network) to simulate a specified function. Learning requires the existence of a systematic procedure for adjusting the network weights to produce that function. (Wasserman, 1989:29)

Learning and representation are thus important and distinct considerations (Hinton, 1989:188-189). For many artificial neural network models, information processing is performed *via* the representation of a function in terms of layers of neurons interconnected by synapses whose strengths (weights) are modifiable--a configuration clearly inspired by neurophysiology. Rosenblatt and others effectively demonstrated that in using such an architectural approach or representation (a symbolic neural network), it becomes possible to discriminate between or correctly classify input patterns, when synaptic weights possess the proper values.

In general, the ability of a network to model or represent functions is thus related to its architecture, which includes neuron layering, synaptic connectivity, various activation and transfer function types, etc., while the process of training the network (i.e. learning) is achieved *via* the application of a rule or algorithm that modifies its synaptic weights sufficiently to allow input vectors to be correctly classified. To clearly understand artificial neural network processing, an analysis of the principal components of a network, including neurons, neural connectivity, and learning rules, is thus required.

2.4.3 Symbolic Neuron Anatomy. Figure 2.3 is a representation of the perceptron, and is offered to illustrate an example of a symbolically modeled biological neuron (Stanley, 1988:84,124; Rogers et al., 1990:49; McClelland and Rummelhart: 1988, 122). This model (or one of its many variants) is fundamental to several neural network architectures used in research. Its basic properties are delineated as follows, in terms of the illustration.

Inputs (x_i) to the neuron enter by way of synaptic connections carrying output signals originating from other neurons or sensor devices. Note there are two numerical components here for each input value, the value itself, produced as output from a sending neuron, and the *weight* (value) of the synapse (w_i) carrying it. An additional input originates from a *bias* neuron, which sends a value fixed at +1 to any receiving neurons; the synapse weight carrying this value (θ), however, is not similarly constrained (it undergoes the usual modification during training). Threshold or bias neurons play different roles in network research; for this study, unless otherwise noted, it is assumed that the bias value is an input value to receiving neurons.

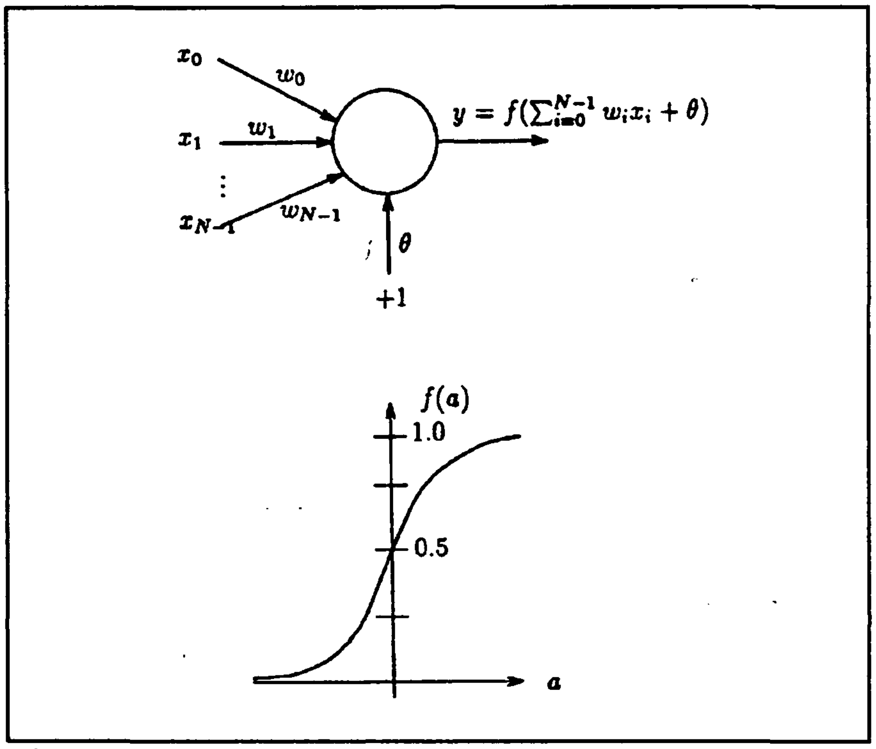


Figure 2.3 The Perceptron (Rogers et al., 1990:49)

To model the total strength of incoming signals on the receiving neuron, all values are passed through an activation function, which often simply transforms them to a weighted sum, $\Sigma(w_i x_i + \theta)$, where each input is multiplied by the weight value of its carrying synapse (Stanley, 1989:84-85). Thus, the activation function determines the response or excitation of the neuron to its total incoming stimuli (input). As indicated in Figure 2.3, the inputs are summed, then passed through a function *f* (discussed next). As an alternative activation function example, some include terms that take the previous activation level of the neuron into account when determining its current activation level, for purposes of modeling local neuronal self-excitation (Stanley, 1990:85).

The neuron's activation value is prepared for export by the application of a transfer function *f*(*a*), (where *a* represents the neuron's activation level), which is often non-linear in form. Many different transfer functions are used in network research, including hard limiter, threshold logic, hyperbolic tangent and sigmoid (or logistic) functions (see Rogers et al., 1990:51; Wasserman, 1989:16-17; Moore, 1990:32). The function below the perceptron in Figure 2.3 is a representation of the output curve for the sigmoidal transfer type, where *a* is scaled from -1 to +1 and *f*(*a*) from 0 to 1. The transfer function transforms the activation value into the neuron's output signal, which is then carried to other neurons in the net *via* synaptic connections.

2.4.4 Error-Driven Learning. For many *supervised* learning techniques, input/output vector pairs (training examples) comprise a training set used for teaching a neural network a classification problem. Learning is said to be supervised in that the set of training examples consists of both the question and the correct answer: the input vector component represents the pattern to be classified, while the output vector component represents the correct (known) classification for its associated input vector. The training set is repeatedly presented to a network to effect learning *via* weight modification: networks are trained to correctly classify input vectors by iteratively forcing the network to compute a classification for the training input vector, comparing this calculation with the correct classification represented by the training output vector, and adjusting network connection weights *via* some measure of difference or error between the two. Weights are adjusted by an algorithm or procedure (learning rule) that attempts to minimize the error between the computed and correct classifications. Weight adaptation may occur with each training presentation on a case by case basis, or after all examples in the training set have been passed through the network. Both the information processing activity of an artificial neuron and the concept of network learning are discussed next, in the context of a specific neural network architecture.

2.4.5 The Perceptron Network. Figure 2.3 in fact illustrates a *network* consisting of a single perceptron. As previously noted, the activation function produces a weighted sum of the inputs plus the bias term, and a sigmoidal transfer function produces the perceptron's output value, which often ranges from 0 to 1 (depending upon the choice of transfer function parameters). The intended function of this network is to classify input training vectors in one of two classes. The perceptron learning procedure is given in Rogers et al. as

> 1. Initialize weights and thresholds to small random numbers. We recommend using a uniform distribution from -0.5 to +0.5.
>
> 2. Present training vector and desired output.
>
> 3. Calculate the actual output:
>
> $$y = f_h\left(\sum_{i=0}^{N-1} w_i x_i + \theta\right)$$
>
> where N is the number of inputs.
>
> 4. Learning:
>
> $$w^{+}_{i} = w^{-}_{i} + \eta(d - y)x_i$$
>
> where η is the gain, $0 \leq \eta \leq 1$, d is the desired output. Note: only *learn* when in error. Alternative learning paradigms include LMS [Least Mean Square] Widrow-Hoff or Gaussian maximum-likelihood weights and threshold selection. (1990:52)

Perceptron learning is thus an iterative process whose ultimate aim is to modify network synapse weights until a state is achieved such that the input training vectors are properly classified by the output classification value computed by the network. In terms of a geometric interpretation for this example, training, in essence

> ...amounts to arranging the weights such that in the region of one class of input objects the output of the perceptron would be near 1; when the features represent the other class of objects, the output of the perceptron would be near zero (or -1 depending on the type of nonlinearity chosen). (Rogers et al., 1990:50,52)

2.4.6 The Linear Separability Problem. The architecture of Figure 2.3 can be extended to a *layer* of perceptrons (additional perceptrons that receive network inputs) (Rogers et al., 1990:52,53). A difficulty known as the "hyperplane limitation problem" (Rogers et al. 1990:53,55) or "linear separability problem" (McClelland and Rumelhart, 1988:123-126) exists for such networks, however, as a consequence of their architecture. For example, given a network consisting of a single perceptron network with two inputs (thus a 2-space input dimensionality), the network will learn to correctly classify inputs only if the classes are separable by a line. For single layer perceptron networks with several inputs (more than 2), the limitation is that classes must be separable by a hyperplane. If this condition does not obtain, the perceptron learning algorithm will not converge (will not sufficiently separate classes into distinct groups after a number of iterations or trials). Two potential difficulties here stem from the possibility that the representation capability of the network's architecture is insufficient for the task at hand, and/or from the possibility that learning algorithm may possess severe limitations and thus require modification.

Due to the restrictions imposed by the single layer perceptron, and the fact that there exist classification problems for two classes *are not* linearly separable, the problem of representation becomes fundamental. The solution requires the addition of a *hidden* node(s) (neurons between the input and output layer) (McClelland and Rumelhart, 1988:123-126). A multi-layer architecture, combined with a variant of the least squares error learning rule (modified for multi-layer network training), makes it possible for neural networks to represent functions in terms of input-output mappings. Prior to discussing the specifics of the solution to the linear separability problem, an analysis of the least squares error learning rule properties is required, as an understanding of its basic form is seminal to the modifications that follow.

2.4.7 Least Squares Error Rule Characteristics. The single-layer (input layer to output) perceptron may use a variety of transfer functions, as previously noted; in the following discussion, a sigmoidal form is assumed. Hinton provides a clear formulation of the least squares learning procedure in his discussion of supervised learning procedures (1989:193-198). Starting with the assumptions of a single-layer network and "...output units whose states (i.e. activity levels) are a continuous smooth function of their total input..." (1989:193), he describes it as follows:

> A measure of how poorly a the network is performing with its current set of weights is
>
> $$E = 1/2 \sum_{j,c} (y_{j,c} - d_{j,c})^2 \qquad (6)$$
>
> where $y_{j,c}$ is the actual state of output unit j in input-output case c, and $d_{j,c}$ is its desired state. We can minimize the error measure given in (6) by starting

with any set of weights and repeatedly changing each weight by an amount proportional to $\partial E/\partial w$.

$$\Delta w_{ji} = -\epsilon(\partial E/\partial w_{ji}). \tag{7}$$

In the limit, as ϵ tends to 0 and the number of updates tends to infinity, this learning procedure is guaranteed to find a set of weights that gives the least squared error. The value of $\partial E/\partial w$ is obtained by differentiating (6) and (1).

$$\partial E/\partial w_{ji} = \sum_{\text{cases}} \partial E/\partial y_j \; dy_j/dx_j \; \partial x_j/\partial y_j$$

$$= \sum_{\text{cases}} (y_j - d_j) \; dy_j/dx_j \; y_i \tag{8}$$

If the output units are linear, the term dy_j/dx_j is a constant. (1989:193-194)

Further, Hinton notes that the geometrical interpretation of this learning rule can be understood in terms of a gradient descent search for a local minimum (1989:194-195):

We construct a multi-dimensional "weight space" that has an axis for each weight and one extra axis (called "height") that corresponds to the error measure. For each combination of weights, the network will have a certain error which can be represented by the height of a point in weight space. These points form a surface called the "error surface"...So gradient descent is still guaranteed to work for monotonic nonlinear input-output functions provided a perfect solution exists. However, it will be very slow at points in weight space where the gradient of the input-output function approaches zero for the output units that are in error.

Hinton thus notes that a perfect solution for the gradient search may not exist (or may not be found within an acceptable time limit). In this case, one approach is to find a solution that yields an acceptable error level, for example, the mean squared error measure between the target classification value and the network's output computation of same.

This is a practical approach, especially when one considers that the technique is highly iterative and computationally intense (Weiss et al., 1991:101-102).

The least squares error approach provides the foundation for the learning algorithm used for training multilayer perceptron neural networks. Upon modification, it provides a means to derive a solution, or at least a solution within the limit of an acceptable error level. What requires discussion next is how it was determined that the addition of hidden neurons (contained in the *hidden* layer between the input and output layers) would overcome the linear separability problem originally posed. Both considerations--the addition of *hidden layer* neurons to aid in representation and a modified learning rule required for training these multilayered networks--are integral to the solution. The general solution is commonly referred to as the backpropagation neural network.

2.4.8 The Backpropagation Neural Network. Given a classification problem possessing non-linear characteristics (here, a two-class discrimination problem), the issue becomes how to construct a neural network representation of the problem that can overcome the linear separability limitation and learn to correctly classify input vectors. The combination of the multi-layer perceptron network and a modified least squares error learning rule provides the solution.

2.4.8.1 Hidden Layer Neurons. A classical problem studied widely throughout the literature is the exclusive or (xor) problem (see Moore, 1990:35-37, for a clear presentation), which was proven intractable (not linearly separable) for a single layer perceptron by Minsky and Pappert in 1969. Hecht-Nielsen notes

> Minsky and Pappert's book *Perceptrons* proved mathematically that a perceptron could not implement the EXCLUSIVE OR (XOR) logical function (f(0,0)=f(0,1)=0,f(0,1) =f(1,0)=1)...,nor many other such *predicate* functions (binary scalar functions of binary vector variables). (1990:17)

With the addition of a *hidden* neuron, the problem becomes solvable. Figure 2.4 illustrates how the two classes in question may be correctly classified if a 3-space plane is applied to the task, and Figure 2.5 illustrates one network architecture that can implement, or *represent*, the solution. By adding a single hidden node (in the hidden layer), the problem is tractable; note that the addition allows the network to discriminate via the addition of a dimension (from 2 to 3-space). In essence, the added input node performs an "and" operation on the first two inputs to create the third coordinate or feature. This enables the network to classify correctly the original 2-space classification problem by separating classes with a 3-space plane (all four possible xor input value pairs are represented in terms of the coordinates of the vertices of 3-space cube shown in Figure 2.4). Thus, for this network solution, the addition of a hidden unit increases the dimensionality of a network to an extent that allows correct xor pattern classification. In their discussion of two-input binary functions, researchers McClelland and Rumelhart note in general that

> ...if you allow a multilayered perceptron, it is possible to take the original problem and convert it into the appropriate three-dimensional problem so it can be solved. Indeed, as Minsky and Pappert knew, it is always possible to convert any unsolvable problem into a solvable one in a multilayer perceptron. (1988:125)

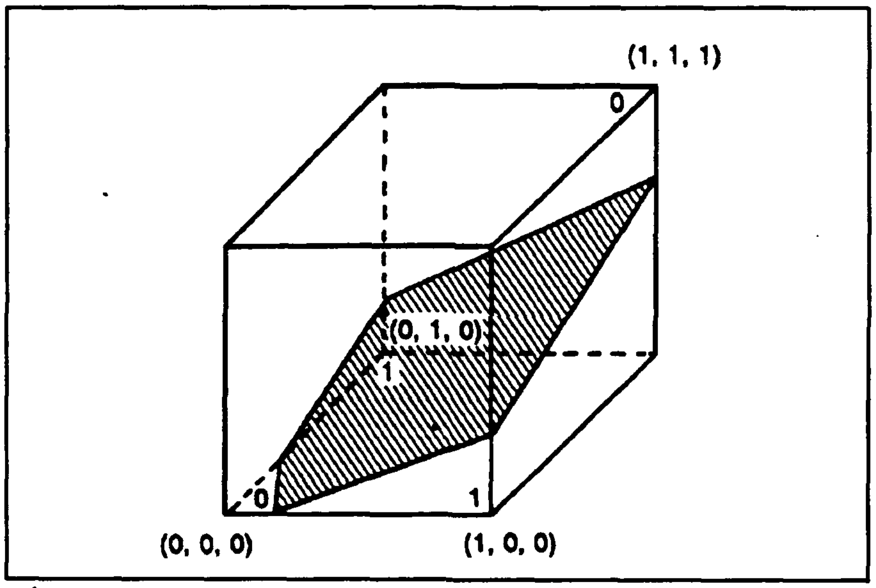


Figure 2.4 XOR Problem Solution
(McClelland and Rumelhart, 1988:125)

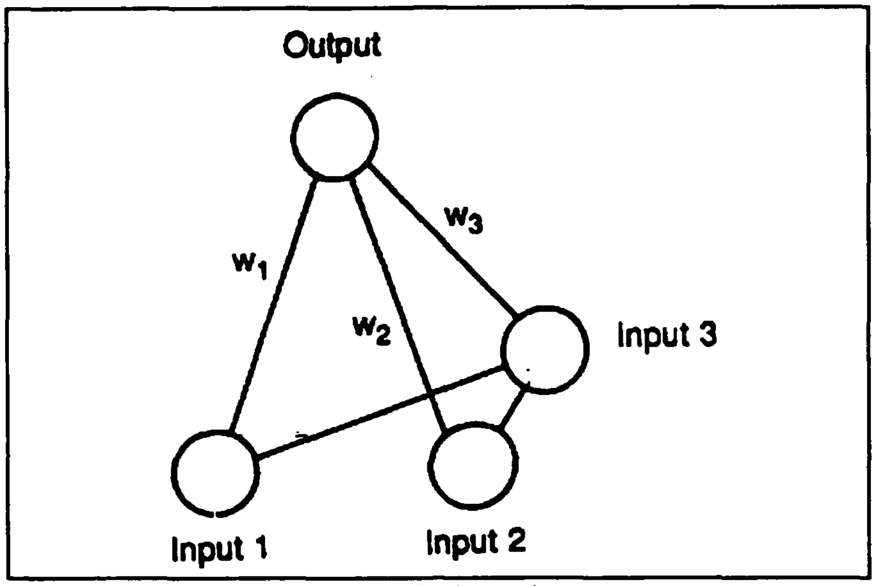


Figure 2.5 XOR Discriminating Network
(McClelland and Rumelhart, 1988:126)

A key point here is that network representation requirements for pattern classification problems can be thought of, to a large degree, a problem of formulating the appropriate multi-dimensional decision regions. In his discussion of the multi-layer perceptron, Moore notes

> Unlike the single-layer perceptron, the multi-layer perceptron can form bounded or unbounded convex decision regions. A bounded convex region means a particular class is contained in a particular finite region while an unbounded convex region contains a class in an infinite region. The convex regions are constructed by the intersections of half planes which are formed by the processing elements in the first hidden layer. Each of the processing elements in the first layer acts like a single perceptron and forms a half plane region bounded by a hyperplane. The decision region for a particular class becomes the intersection of all the half planes that are formed be each of the processing elements. (1989:37-38)

Moore additionally notes that the shape of the decision region is contingent upon the type of transfer function employed:

> The shapes of the decision regions can change depending on the types of transfer functions used by the processing elements. When sigmoidal nonlinearities are used instead of hardlimiting nonlinearities, the decision boundaries are curved instead of straight line segments. Networks form these decision regions by using the backpropagation training algorithm (17:16). (1989:37-39)

2.4.8.2 The Backpropagation Learning Rule. As suggested by McClelland and Rumelhart, multi-layer perceptron networks are indispensable for problem representation. Yet such architectures create another difficulty, as network training algorithms must modify weight synapses for *additional* (hidden) layers of neurons. Hinton describes the backpropagation of errors learning solution for this problem as follows:

> The "backpropagation" learning procedure...is a generalization of the least squares procedure that works for networks which have layers of hidden units between the input and output units....In a multi-layer network it is possible, using (8) [as cited above], to compute $\partial E/\partial w_{ji}$ for *all* the weights in the network provided we can compute $\partial E/\partial y_j$ for all the units that have modifiable incoming weights....The central idea of backpropagation is that these derivatives can be computed efficiently by starting with the output layer and working backwards through the layers. For each input-output case, *c*, we first use a forward pass [using weighted sum activations and sigmoidal transfer functions], starting at the input units, to compute the activity levels of all the units in the network. Then we use a backward pass, starting at the output units, to compute $\partial E/\partial y_j$ for all the hidden units. For a hidden unit, *j*, in layer *J* the only way it can affect the error is via its effects on the units, *k*, in the next layer *K* (assuming units in one layer only send their outputs to units in the layer above). So we have

$$\partial E/\partial y_j = \sum_k \partial E/\partial y_k \; dy_k/dx_k \; dx_k/dy_j$$

$$= \sum_k \partial E/\partial y_k \; dy_k/dx_k \; w_{kj}, \qquad (9)$$

where the index *c* has been suppressed for clarity ... Notice that the computation performed during the backward pass is very similar in form to the forward pass (though it propagates error derivatives instead of activity levels, and it is entirely linear in the error derivatives). (1989:198-199)

With the multi-layered neural network and backpropagation learning rule combination, many function classification mappings thought to be intractable are provided a solution. The implementation algorithm for backpropagation is given by Rogers et al as

1. Initialize weights and thresholds to small random numbers. We generally use a uniform distribution from -0.5 to +0.5.

2. Present training input and classification (desired output).

3. Calculate output. [the forward pass]

4. Learn (adapt weights and thresholds) [the backward pass]

$$w^{+}_{ij} = w^{-}_{ij} + \eta\delta_j x_i + \alpha(w^{-}_{ij} - w^{--}_{ij})$$

where w_{ij} is the weight from node *i* to node *j* in the next layer, x(i) is the output of node *i*, and δ(j) is the *error* associated with node *j*. η and α are learning rates ... w^{+}_{ij} is the new weight value and w^{-}_{ij} is the old weight value. w^{--}_{ij} is the value of the weight before the last update. Thresholds are adapted similarly where x_i is replaced by +1 if the threshold is *added* to the weighted sum and -1 if it is *subtracted*. The η_j are defined as follows:

$$\eta = \begin{cases} y_j(1-y_j)(d_j-y_j) & \text{for output node } j \\ x_j(1-x_j)\sum \eta_k w_{jk} & \text{for hidden node } j \end{cases}$$

where d_j is the desired output for output node j and y_j is the actual output. (1990:59)

2.4.8.3 Backpropagation Networks as Function Mappings. A key concept for this research lies in understanding that the backpropagation network is, in essence, a generalized non-linear (note that sigmoid transfer functions are typically employed) *mapping* function (between its input and output vectors) formulated by successive (iterative) refinement *via* a gradient descent heuristic. Hecht-Nielsen notes

> The information processing operation that backpropagation networks are intended to carry out is the approximation of a bounded mapping or function $f\colon A \subset \mathbf{R}^n \rightarrow \mathbf{R}^m$, from a compact subset A of n-dimensional Euclidean space to a bounded subset $f[A]$ of m-dimensional Euclidean space, by means of training on examples [*input/output vector pairs*] $(\mathbf{x}(1),\mathbf{y}(1))$, $(\mathbf{x}(2),\mathbf{y}(2))$, ...$(\mathbf{x}(k), \mathbf{y}(k))$,... of the mapping, where $\mathbf{y}(k) = f(\mathbf{x}(k-))$. As always, it will be assumed that such examples of a mapping f are generated by selecting $\mathbf{x}(k)$ vectors randomly from A in accordance with a fixed probability density function $\rho(x)$. The operational use to which the network the network is to be put after training is also assumed to involve random selections of input vectors $\mathbf{x}$ in accordance with $\rho(x)$. (1990:125)

From this perspective, a backpropagation neural network is essentially a non-linear function approximation technique that maps an n-space input vector to an m-space output vector. To emphasize, note that the input/output vector component values need not be binary (0,1):

> Many applications use bit representations (0,1) for symbols, and attempt to have a neural net learn fundamental relationships between symbols...There is no fundamental reason, however, to use integers as values for Input and Output. If the Inputs and Outputs are

instead a collection of floating point numbers, then the network, after training, yields a specific continuous function in n variables (for n inputs) ... that provides a type of nonlinear, least mean square interpolant formula for the discrete set of data points in the training set. (Lapedes and Farber, 1987:4)

Thus, backpropagation nets can map both boolean and floating point (real) input/output vectors. At a fundamental level, the representation these networks accomplish is essentially one of two basic types: in the case of mapping symbolic (0 and 1-valued) input vectors, the networks map the vertices of an n-dimensional hypercube (where n is the number of input nodes) to their output(s); for continuous (real) valued data, they form a multi-dimensional curve from which output values are interpolated or extrapolated. Note that in the case of boolean (symbolic) mappings (0-1 hypercube vertices as input), there is no sense in which the network provides computed outputs *via* interpolation--the networks are solely provided the coordinates of hypercube vertices from which to accomplish an associated output mapping.

Hecht-Nielsen attributes the first insight in determining "...what functional forms can be approximated by neural networks..." (1990:131) to the discovery of Kolmogorov's theorem:

In 1957 Andrei Kolmogorov published an astounding theorem concerning the representation of arbitrary continuous functions from the n-dimensional cube $[0,1]^n$ to the real numbers R in terms of functions of only one variable... (1990:122-123).

Stated in terms of a neural network, Hecht-Nielsen gives this theorem as

> **THEOREM 1** Given any continuous function $f : [0,1]^n \rightarrow \mathbf{R}^m$, $f(x) = y$, f can be implemented exactly by a three-layer feedforward neural network having n fanout processing elements [inputs] in the first ($\mathbf{x}$ - input) layer, (2n+1) processing elements in the middle layer, and m processing elements in the top ($\mathbf{y}$ - output) layer (1990:122)

In discussing the parameters and functional forms required to actually implement this theorem, Hecht-Nielsen notes that no method of construction has been devised. Kolmogorov's theorem is an existence theorem--it guarantees a solution *exits* for certain function mappings, but it tells us nothing about the functional forms or constants required for the actual realization of any function mapping. It did, however, provide the foundation for an important result directly related to the backpropagation network. Noting the independent discovery of empirical results that led to the development of the *backpropagation approximation theorem* (Theorem 2 below), Hecht-Nielsen explains the conditions of its genesis, its guarantee, and its limitation:

> Kolmogorov's theorem was the first step... the back-propagation neural network is itself able to implement any function of practical interest to any desired degree of accuracy...Given a function $f : [0,1]^n \rightarrow \mathbf{R}^m$, we say that f *belongs to* L_2 (or "*is* L_2") if each of f's coordinate functions is square-integrable on the unit cube. For functions of this class it is assumed that the $\mathbf{x}$ vectors are chosen uniformly in $[0,1]^n$ (relaxing this condition is easy)...
>
> **THEOREM 2** Given any $\epsilon > 0$ and any L_2 function $f : [0,1]^n \rightarrow \mathbf{R}^m$, there exists a three-layer neural network that can approximate f to within ϵ mean squared error accuracy....

...Finally, although the above theorem guarantees the ability of a multilayer network *with the correct weights* to accurately implement any arbitrary L_2 function, it does not comment on whether or not these weights can be *learned* using any existing learning law. That is an open question. (1990:131-133)

2.4.8.4 Backpropagation Network Generality.

The multi-layer perceptron trained with the backpropagation learning rule, commonly called the backpropagation neural network, constitutes an important development in the history of neural network research:

The backpropagation neural network is one of the most important historical developments in neurocomputing. It is a powerful mapping network that has been successfully applied to a wide variety of problems ranging from credit application scoring to image compression. (Hecht-Nielsen, 1990:124)

Specific examples of its application are accordingly presented next, in a review of neural network research literature.

2.5 Previous Neural Network Research

The following review of neural network research focuses on studies in which the effectiveness of backpropagation was compared with that of more traditional approaches, and where fundamental issues of network problem representation are discussed in detail. The reader should note that a wealth of literature is available on neural network research, spanning a vast array of academic disciplines. In the following review, four essential topics are covered, one per review. In the work of Fishwick (1989) and Orris and Feeser (1990A), traditional analysis approaches are compared with backpropagation networks. Lapedes and Farber offer recommendations for network architecture, and Sanger applies principal component analysis to the problem of understanding how a 3-layer backpropagation network solves a problem.

2.5.1 Fishwick (1989). In his study of the behavioral modeling characteristics of a neural network, Fishwick compares backpropagation network, linear regression and surface response methodology techniques. Choosing a ballistics model for testing, a network (three input nodes, two hidden layers of 5 nodes each, and one output node) was constructed to model the horizontal distance traveled (*h*) by a projectile, given the initial *conditions* of angle, velocity and height. The network was trained for 180,000 iterations and tested by requiring it to predict values of *h* for specified initial angle and velocity values (previously tabulated by applying differential equations of motion). Fishwick notes

> We learned that the neural network was a relatively poor approximation method given the constraints of our experiment....Simple linear regression outperformed a neural network (with root mean square error (RMSE) of 46.47 versus the neural net's of 70.85). The surface response method fared even better with a RMSE of 29.19. (707-708)

An important theme stressed throughout this study lies in the importance of assessing *how much is known* about a system in order to select an appropriate technique for modeling:

> Even when we appear to have little knowledge about a system, we usually have enough to assume a basic system structure (often a canonical form) and then we can estimate parameters" (709).

In contrast to the more traditional analysis methods, Fishwick hypothesizes that the primary reason why neural networks appear to be inadequate is that they do not "capture the system structure characteristic of all physical models" (702).

2.5.2 Orris and Feeser (1990A). In their comparison of neural networks and linear regression analysis, the authors found that the former produced R^2 values almost as good as those produced by the latter. Three tests were performed using traditional, simulated, and actual data.

The traditional Longley test data, "...used extensively for testing the computational accuracy of regression algorithms since the data has a high degree of multi-colinearity" (4), was modeled first *via* regression analysis, which exhibited a multiple R^2 of .99548. The maximum coefficient of determination attained by a neural network model of same was .98901. It is noted that the network R^2 values were computed (for all tests in the study) as follows:

> The predicted values were correlated with the target values and the square of this value corresponds to the multiple R^2 of multiple regression analysis, which is interpreted as the amount of variance accounted for in the dependent variable. (3)

To assess the extent to which a network could detect a "hidden pattern" in data, a single sample containing observations representing two distinct relationships (intersecting ellipses with positive and negative orientations) was constructed. The two independent variables were "group" (a dichotomous variable indicating to which ellipse the observation belonged) and "X" (a metric x-axis coordinate); the dependent "Y" variable was a (metric) y-axis coordinate. The linear regression R^2 value was .015 while the network exhibited a coefficient of determination of .85. The authors acknowledge that "Since the linear regression line tries to accommodate the entire data set, the regression line is almost flat..." (4). The authors report that while neural networks provided relatively good predictions of the test data, there was no immediate way to glean an *explanation* from same for understanding the relative importance of variables in the model.

In the third and final test, a network was tested using actual data relating Earnings per Share (EPS), Price/Earnings Ratio (P/E), Return on Equity (ROE), Sales per Employee (S/EMP), Growth in Stock Price and Standard Industry Classification (SIC) Code to an independent variable measuring corporate environmental responsibility (5). The network coefficient of determination was .749 compared to .146 for the corresponding regression analysis. They state

> The results from the actual data indicated the neural network was accounting for up to four times as much variability in the dependent variable. The bad news, from an explanatory point of view, is that we have no idea why. (6)

In summary, Orris and Feeser state "The main conclusion is that it appears that neural networks do not appear especially useful for routine use in exploratory data analysis" (7).

2.5.3 Lapedes and Farber (1987). In their work on modeling chaotic time series, Lapedes and Farber report that "..for certain applications neural networks achieve significantly higher numerical accuracy than more conventional techniques" (1). The authors state

> The Glass-Mackey has a strange attractor with fractal dimension controlled by a constant parameter appearing in the differential equation. We present results on a neural network's ability to predict this system at two values of this parameter, one value corresponding to the onset of chaos, the other value deeply in the chaotic regime. We also present the results of more conventional predictive methods and show that a neural net is able to achieve significantly better numerical accuracy. (2)

While a detailed discussion of this experiment is beyond the scope of this research, Lapedes and Farber present several additional points regarding network architecture that suggest a methodological approach, and therefore require mention.

A distinction is made between the architectures required for networks used in symbolic processing and that required to process floating point (real-valued) inputs and outputs. In regard to the latter, the authors note

> One does not need more than two hidden layers for processing real valued input data, and the accuracy of the approximation is controlled by the number of neurons per layer, and not the number of layers. We emphasize that although two layers of hidden neurons are **sufficient** they may not be **efficient**. Multilayer architectures may provide very efficient networks (in the sense of number of neurons and number of weights) that can perform **accurately** and with minimal cost. (14)

In the case of symbolic input and output, the authors note that while it is entirely feasible to model this problem type with two hidden layers, it is not appropriate for forecasting new values: "This is an effective method for **memorizing** the training set, but a very poor method for obtaining **correct** predictions on new input data" (13). Through a geometric analysis, they conclude that one hidden layer of neurons is more appropriate for modeling symbolic relationships (12).

2.5.4 Sanger (1989). Sanger's research focuses on the inner workings of neural networks, in an attempt to understand how a 3-layer (18 input, 19 hidden, and 21 output nodes) network solves a particular problem. Using a method the author entitles *contribution analysis* for "deriving the responsibilities of individual hidden neurons in implementing the input-output mapping" (115), Sanger investigates a scaled-down version of the NETtalk neural network, which "...learns to convert written English text to the corresponding spoken English phonemes" (116).

Sanger defines *contribution* as follows:

For a specific input presentation, a specific hidden unit, and a specific output unit, the contribution is defined as the product of the hidden unit's activation when the net is presented with the specified input and the weight from the hidden unit to the output unit. There is a distinct contribution for each combination of input presentation, hidden unit and output unit. (116)

The network is trained by presenting two-letter combinations to the net together with the correct phoneme (only two input nodes and one output node are "on" (set to 1 instead of 0) in a given training example (presentation)). Sanger then analyzes the distribution of contributions produced by each output and hidden unit, *via* principal component analysis. With respect to output units, "... the principal components represent patterns of hidden units that are responsible for activating the specific output unit...", while for each hidden unit "...the principal components represent the patterns of output units that the specific hidden unit is responsible for..." (120).

For collective hidden node responsibilities, it is noted that a small number of hidden units usually contribute strongly to exception processing (rare input-output combinations), that membership in hidden unit patterns is well-defined (for input-output relationships recurring often), and that "..the net learns which output units can be handled identically and thus consolidates the responsibility for identical units" (122).

Individual hidden unit responsibility analysis reveals that "a strong dichotomy exists between units that are responsible for vowels and those that are responsible for consonants", and "the net appears to allocate more resources toward learning exceptions than toward learning [general] rules" (125).

Finally, Sanger experiments with the number of nodes comprising the hidden layer and summarizes his findings by succinctly noting "as hidden nodes are added to a net, the proportion of superfluous units increases" (128).

2.5.5 <u>Research Literature Summary</u>. The work of Fishwick suggests that it is critical to assess how much is known about a problem before one selects an appropriate modeling technique, and that it appears that networks cannot represent system structure characteristics of physical systems. Orris and Feeser's research suggests that networks may be very good predictors, but divulge little about variable significance. Lapedes and Farber note that for their research, a neural network was a superior time-series predictor, while Sanger suggests that it is possible to understand how a network works *via* principal component analysis.

Thus, the analysis of neural network applicability appears to be highly problem specific, suggesting that much additional research is required for understanding how to accurately represent and analyze a problem. The methodology for the problem under analysis in this research is presented next in Chapter III; it defines an experiment designed to assess network predictive accuracy in comparison with that of multiple linear regression models. Subsequently, the presentation of the analysis and findings is presented in Chapter IV.

III. Methodology

Methodology Overview

This research fits twelve backpropagation neural network metamodels to attack and no-attack scenario data representing the first six days of a one month time period as previously generated by Diener *via* simulation (1989). A neural network metamodel and a fully specified linear regression counterpart are fitted to each daily sample, for each of the twelve days under study, and compared on the basis of predictive accuracy. The fully specified (two-way interaction) regression metamodels are not offered as the product of an attempt to determine the best linear model (*via* PRESS residuals, C(p)-Mallows statistics, etc.). Likewise, the neural networks are basic 3-layer (input, hidden, output) backpropagation models that are conservatively built, trained and tested; advanced network techniques (training with noise, synapse pruning, multiple hidden layers, etc.) are intentionally avoided in this research in an attempt to create fundamentally basic and identically constructed neural networks for comparison with their regression counterparts. While the focus of this research is indeed prediction, the point in comparing unaltered and basic models in lieu of models specifically selected for their predictive capabilities lies in the attempt to more fully understand the fundamental differences between the prediction methods. An additional rationale for this limitation lies in the desire to explore the extent to which a 3-layer neural network can accurately forecast sorties flown solely on the basis of ten input variables (resource factors set at "low" and "high" indicators represented as 0 and 1 respectively), in comparison to regression.

Predictive accuracy is assessed for both model types (regression and network) *via* the use of an independent testing sample not used in the model fitting process. The fitting (or training) and testing samples contain 128 and 20 observations, respectively. Predictive accuracy is assessed *via* Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and other measures. Test results are summarized and interpreted in Chapter IV.

3.1 Sample Considerations

3.1.1 Simulated Data.

All neural network and regression metamodels examined in this research are developed and analyzed *via* simulated data samples. While the requirement to use simulated data is largely inherited due to the fact that this research is a follow-on study of specific aspects previously noted and examined in Diener's research (1989), certain points regarding the use of simulated in lieu of "live" data require mention.

In comparison with the obviously impractical idea of interrupting air base operations to study the effects of policy changes on sortie generations, simulation offers a viable alternative. However, to appropriately assess the use of simulation for data generation, one must first possess a generally clear notion of what simulation entails:

> Simulation is a form of modeling whose purpose is usually comprehension, planning, prediction, and manipulation. It can be defined broadly as a behavioral or phenomenological approach to modeling; that is, a simulation is an active, behavioral analog of its referent. The essence of simulation is that it unfolds over time. It models sequences and (possibly) timings of events in the real world. Simulation is a *process* in which a model of *any* kind is used to imitate (some aspect of) the behavior of its referent. It denotes an action (process) rather than a thing.
> (Rothenberg, 1989:8)

Simulation can thus be characterized as a process model that attempts to capture specific aspects of the temporally-based behavior of its referent. For this research, the referent is an F-15 air base, the process being modeled is that of sortie generation, and the specific behavioral aspect under study is the dynamic relationship between resource factors and sorties flown--the inputs and outputs to the process being modeled--as it unfolds throughout the first six days of a thirty day simulation period previously studied by Diener (1989).

In regard to assessing the quality of a simulation model, the concepts of verification and validation are fundamental. Verification is primarily a measure of the system's ability to perform according to a specification (often a software design); while validation entails a much broader and often exceedingly complex range of considerations. While the reader is referred to the literature for in-depth treatment of the issue of validation, two fundamental concerns are assessing the extent to which the simulation is isomorphic to real-world conditions, and determining its level of credibility among the consumers of its products. For this study, the latter concern is largely answered in terms of the utility of TSAR and TSARINA to researchers investigating the behavior of large-scale systems. In regard to the former, it is noted that any serious systems modeling tool undergoes a process of evolutionary improvement, and that model validation, in this case, falls under the purview of the RAND corporation. On the basis of the intrinsic value of TSAR and TSARINA for large-scale systems investigation and its use in previous research (notably in Diener (1989)), it is argued that the simulation model is sufficiently reliable for data sample generation.

3.1.2 Randomness and Representativeness. The issue of the representativeness of Diener's (1989) samples (used for model fitting) is addressed by noting that the original treatment selections comprising the sample are contingent upon the fractional factorial design construction process selected by Diener (1989) (as previously noted, the specific treatments comprising each sample used for fitting are listed in McLean and Anderson in fractional factorial design plan 8.10.16 (1984:255-256)). The experiment performed in this study assumes that the sample observations are real-world "control-group" data--not estimates of the *expected* values of same, and that the orthogonal properties of the fractional design employed ensure balance of representation of each main effect *via* internal factor replication implicit in the construction of the design matrix itself (internal replication is not to be confused with the replication of *treatments*). Thus it is assumed that the plan employed produces samples representative of the underlying population probability distribution to the extent that significant main effects and two-way interactions existing in the population can be readily detected from the generated samples.

The representativeness of testing samples is also addressed in terms of Diener's (1989) experimental design: each randomly selected testing treatment is first assigned a randomly generated block number, where the set of block numbers employed consists of only those previously used in the original (fitting) sample generation process (from the experimental design employed by Diener). The block assignments control which random number streams are used, thus the introduction of "new" randomness (in comparison with the training samples) is restricted. The simulation (TSAR/TSARINA) generates twenty response values for the treatments selected for testing.

To sum: with respect to the control of the impact of model variance on response values (sorties flown), both fitting and testing samples are subjected to the original variance reduction technique (VRT) previously employed by Diener (1989) to ensure that initial experimental conditions are not violated.

3.2 Multiple Linear Regression Metamodel Methodology

The regression metamodels used in this research are fully specified, including terms for each of the 10 independent variables, all possible two-way interactions of same, with no blocking term. The metamodel form for both the no-attack and attack scenarios is essentially given by Eqs (2.4) and (2.5), respectively. As stated above, *no* attempt is made to reduce the full model given by those forms (*via* variable elimination techniques), as the intent is to compare the empirical behavior of two different modeling methods. Diener's (1989) original methodology uses backward elimination to reduce the number of independent variables present in the attempt to identify those models with high *explanatory* capability. As previously noted, there is a fundamental difference between selecting models on the basis of explanatory capability (in terms of how well they fit a given sample), and selecting models exhibiting the best *a priori* predictive potential (see *2.3.6*). Thus, Diener's reduced metamodels (1989), selected for their explanatory capabilities, are not used for the comparisons performed in this follow-on study. Further, as the focus of this research is *prediction comparison*, the regression metamodels and their network counterparts are left unaltered to facilitate exploration of the their fundamental differences in predictive performance.

All regression metamodels are developed using the *Statistical Analysis System (SAS)* software running on a VAX/VMS Minicomputer hardware platform. For testing, each model is required to predict sorties flown values for each observation contained in each (daily) testing sample.

3.3 <u>Network Metamodeling Methodology</u>

The success or failure of backpropagation neural network applications is highly contingent upon several empirical factors. Considerations that must be taken into account include the number of hidden layers used, the possibility that the network may be *overtrained* and/or *overspecified* (both hamper predictive capability or "generalization"), the effects of different initial random weight settings, and the possibility that the gradient search may become trapped in a local minimum existing in the error surface (and thus not "find" the global minimum error) (Hecht-Nielsen, 1990:115-119, 128-131). Such factors are application-specific. Accordingly, the initial conditions of the experiment are reviewed next.

3.3.1 <u>Network Modeling Environment</u>. All neural network metamodels analyzed in this research are developed using *BrainMaker Professional, Version 2.13,* a commercial computer software package designed and developed by California

Scientific Software (California Scientific Software, 1990; California Scientific Software Technical Support, 1991; Lawerence and Lawerence, 1990). With respect to hardware, all neural network models are trained and tested on an IBM-PS2 compatible personal computer configured with an 80386/-80387 chip set rated at 25 MegaHertz.

3.3.2 Network Architecture. The architecture used in every neural network model in this research is limited to the input layer, a single hidden layer, and the output layer: An input layer of 10 nodes represents nine resource factors and the environmental variable of aircraft attrition (the independent variables) (see Tables 2.1 and 2.2), one-output node represents the response variable sorties flown (the dependent variable), and an empirically determined number of hidden nodes comprises the hidden layer (which is determined by adjusting their number to allow the network to learn the fitting/training set to within a pre-specified error tolerance measure). A model is developed for each of the twelve days under analysis (the first six days of both Attack and No-Attack Scenarios).

3.3.2.1 Initial Design. The careful reader will note that neither the regression metamodels nor the networks contain a blocking term. The ommission in the network case is prompted for the following reasons:

1. At the general level, we desire to see how well the networks fare in capturing the input/output mapping *via* a

simple and intuitive modeling approach (10 resource and policy factors as input, 1 sorties flown response variable output).

2. The most immediate way to represent blocking to the networks seems to require the addition of eight additional input nodes (a binary string of length eight), which increases the size of the input vector to eighteen components. The number of hidden nodes in a (one hidden layer) network "drives" the number of weights (synaptic connections) multiplicatively, thus there is a greatly increased potential for the network to become *overspecified* in relation to the size of the training samples (in network modeling terms, the number of weights or free parameters in the network may quickly exceed the number of training examples (128 per daily metamodel)).

While the issue of model overspecification is more directly addressed in *3.3.2.2* below, consider the following brief example: given 18 in and 1 out, a possible hidden layer consisting of 5 hidden nodes would create a network containing 114 weights: 18*5 + 18 (bias connections) = 108 connections from input to hidden layers, 5*1 + 1 (bias) = 6 from hidden to output layers, for a total of 114 (simply 108+6). Comparing the size of this possible representation (18 inputs) to the number of weights a 10-input network would generate (at 5 hidden nodes the network would contain 61 weights) suggests that the number of free parameters (weights) in the large network may grow too large in terms of an initial design decision, hence the 10-?-1 architecture is selected for preliminary investigation (where *?* denotes an as-of-yet unspecified number of hidden nodes) over the 18-?-1 model.

The need for this design conservatism is further amplified when considering that each model's training (fitting) sample contains only 128 observations, and, similar to regression, the danger of *overfitting* the model via the inclusion of an excess number of free parameters (weights for networks, beta coefficients for regression) is a real one.

3. As an ancillary research question, it is desirable to investigate the effect the blocking variable has on response forecasting. In the event that neither metamodel type does well at predicting the testing samples, the preliminary conclusion would appear to be that the representation of blocking is important for controlling response variance (regardless of metamodel type). If nets fare well but regression does not, the implication may be that a neural network modeling technique may compensate for the ommission of the blocking term *via* specific modeling properties. By ommitting the blocking variable from the metatmodels, the opportunity exists to explore this research issue.

3.3.2.2 Hidden Layer Nodes. Determining the required number of hidden layer nodes for effective input/output mapping is often a non-trivial task (it is essentially an attempt to find an appropriate model architecture specification for the problem at hand). It is possible to over/under-specify the network model in a manner analogous to regression model specification. If too few hidden nodes are used, the possibility exists the network either will not converge or sufficiently learn enough of the underlying problem structure to generalize (predict) well; if too many hidden nodes are employed the network may "memorize" the training set and exhibit "look-up table" characteristics, with prediction inaccuracy a probable consequence.

As the weights (synaptic connections) of a network are its independent variables (or, equivalently, the free parameters of the gradient descent technique employed to determine the minimum fitting error), caution must be exercised to ensure that the number of training examples exceeds their total number. With regard to the relationship between the number of weights in a network and generalization capability, Hertz et al. note

> Another important lesson about generalization can be learned from statistics and curve-fitting; too many free parameters results in **overfitting**....a curve fitted with too many parameters follows all the small details or noise but is very poor for interpolation or extrapolation [prediction]. The same is true for neural networks: too many weights in a network give poor generalization. (1991:147)

Although a definitive formula for calculating the correct number of weights for a specific application remains an issue for research (Rogers et al., 1991:64-65), it has been recommended that the number of training examples should be between 2 and 10 times the total number of nodes (neurons) contained in the network--as a practical rule of thumb (California Scientific Software, 1991).

The reader is cautioned with regard to directly comparing the number of independent variables in a network topology with same for a regression model or non-linear equation matrix form as the primary focus of much of neural network research entails pattern recognition/learning in a data-driven environment in lieu of statistically precise mathematical forecasting methods such as regression. The calculation of future values (prediction) in the traditional mathematical sense (especially *via* regression) is a thoroughly researched field with well-defined and robust procedures, while in the case of neural networks, research remains highly empirical. The difference, in theoretical terms, entails comparing Kolmogorov's technique for approximating continuous functions with a Taylor's polynomial approximation for same. It is far from impossible that the number of independent variables or nodes may be larger or smaller than suggested by the guidelines noted above, depending on the type of network, representation complexity, sample size constraints, etc. In this research the number of nodes (and therefore the number of independent weight variables) is determined by limiting the network topology to 1 hidden layer and empirically testing whether or not the network will converge or reach a pre-specified error tolerance at different numbers of hidden neurons.

3.3.2.3 Synaptic Connectivity. With respect to synaptic connectivity, each network model is *feedforward* in design, as depicted in Figure 3.1. Hertz et al. provide a clear description of this form of network architecture:

> There is a set of input terminals whose only role is to feed input patterns into the rest of the network. After this come one or more intermediate layers of units, followed by a final output layer where the result of the computation is read off....there are no connections leading from a unit to units in the previous layers, nor to units in the same layer, nor to units more than one layer ahead. Every unit (or input) feeds only to the units in the next layer. The units in the intermediate layers are often called *hidden units* because they have no direct connection to the outside world, neither input nor output. (1991:90)

In terms of Figure 3.1, each node in a given layer is connected (by synapses) to every node contained in the layer immediately above it, no connectivity between nodes occupying the same layer is permitted, and no node (for example, in layer *j*) may be vertically connected to a node existing in a layer other than the one immediately above (*k*). For this research, this choice of architecture, a basic 3-layer feedforward neural network employing the backpropagation learning rule) is, in part, inspired by Hecht-Nielsen's backpropagation approximation theorem cited in *2.4.8.3*.

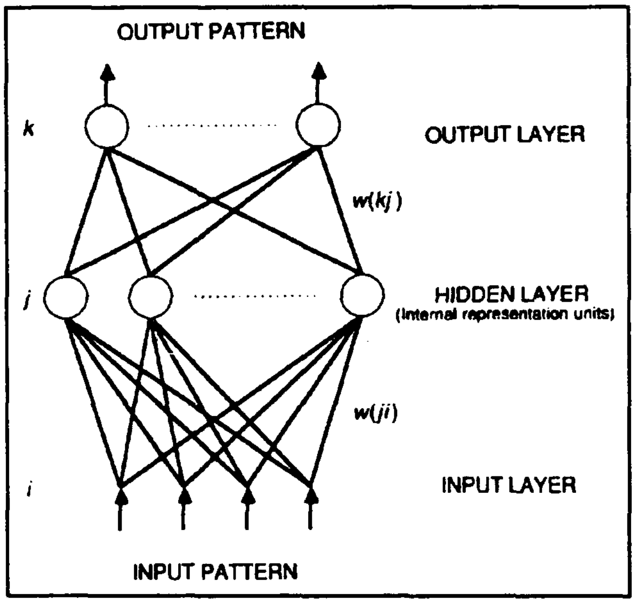


Figure 3.1 Feedforward Network
Pao, 1989:121

The feedforward architectural form is stable in that it is not subject to feedback oscillation between nodes or layers, which feedback (or recurrent) neural networks exhibit or deliberately model (for the purpose of inducing simulated annealing, adaptive resonance, etc.). The reader should note that the backpropagation training algorithm uses forward and backward passes to compute outputs and adjust weights based on computed error (see *2.8.4*), respectively, which is not at all equivalent to inducing feedback between nodes/layers in the attempt to force the network reach an energy minimum (which is essentially a different training technique). For additional information on feedback and other alternative architectures and training methods, the reader is referred to Hecht-Nielsen (1990), Stanley (1990), and Hertz et al. (1991).

3.3.3 Hidden Layers. The number of hidden layers required to accommodate the effective learning of input-output mappings, while empirical, entails the notion of input transformation: as input signals propagate through successive layers in the network, they are transformed into patterns that are hyperplane-separable and therefore amenable to network classification (see *2.4.8*). As previously noted (*2.5.3*), binary networks (binary inputs and outputs) appear to generalize best when only one hidden layer is used, while it is (theoretically) possible to model most all continuous-valued problems with two hidden layers. In the interest of exploratory research and on the basis of the Hecht-Nielsen's backpropagation theorem (*2.4.8.3*), this study limits the number of hidden layers (for any network developed herein) to one.

3.3.4 Network Learning Rule. The backpropagation of errors learning rule is employed to train all network models developed in this research. The specific form of the learning algorithm used in this research, implemented in the *Brainmaker Professional* software (1990), is discussed by Stanley as follows.

> Basically training consists of running patterns through the network forwards, then propagating the errors backwards, and updating the weights according to the equation
>
> $$D_p = \eta(\delta_{pi}O_{pj}) \qquad (3.1)$$

where η is known as the "learning rate". In Brainmaker's actual implementation, we use a version of the rule used by Sejnowski and Rosenberg in their NetTalk application, whereby

$$D_pW_{ij} = \eta((1-\mu)\delta_{pi}O_{pj} + \mu D_{p-1}W_{ij}) \qquad (3.2)$$

Here, μ is another parameter known as a "smoothing factor". It improves convergence [a gradient descent search for the global minimum] somewhat, but even if μ is set to 0, the algorithm will still converge [if a solution exists], although it will take slightly longer. (1990:241)

This research uses the learning algorithm listed above as Eq(3.2). As discussed by Stanley (1990:237-242), the components of Eq(3.2) are defined as

D_pW_{ij}	=	the change in weight W_{ij} on pattern *p*,
δ_{pi}	=	$-(\partial E_{pi}/\partial O_{pi})O_{pi}$ (see *2.4.7* and *2.4.8.2*),
O_{pj}	=	the output of neuron *j* on pattern *p*,
η	=	the learning rate, and
μ	=	the smoothing factor [or momentum term].

In their discussion of momentum terms, Hertz et al. note

...gradient descent can be very slow if η is too small, and can oscillate wildly if η is too large...The idea is to give each connection w_{pq} some inertia or momentum, so that it tends to change in the direction of the *average* downhill "force" that it feels, instead of oscillating wildly with every little kick. Then the effective learning rate can be made larger without divergent oscillations occurring....the addition of a **momentum term** [Plaut et al., 1986], is often effective and is very commonly used...a value of 0.9 is often chosen. (1991:123)

The momentum term referred to is essentially the same entity as the smoothing factor use in *Brainmaker*, shown in Eq(3.2).

For this research, the smoothing factor (or, equivalently, momentum rate) is left unaltered at μ=.9, to take advantage of its beneficial effects on higher rates of learning (for this research, at η=1 and .5).

3.3.5 Transfer Function. The transfer functions (TFs) used in all networks developed this research are sigmoidal in form. As implemented in *Brainmaker Professional* (California Scientific Software, 1990), it is given by Stanley (1990:242) as

$$TF(A) = [(High-Low)/(1+\exp(-Gain*(A-Center)))] + Low \quad (3.3)$$

where A represents the activation value of the neuron or node under consideration (see *2.4.3* for a description of symbolic neuron anatomy, including transfer functions). Further, Stanley notes (1990:242-243)

> If we set High to 1, Low to 0, Gain to 1, and Center to 0, this reduces to the form used by Rumelhart:

$$TF(A)=1/(1+\exp(-A)) \quad (3.4)$$

This study utilizes the transfer function defined in Eq(3.4). It should be noted that sigmoidal transfer functions are used only in the hidden layer nodes--input and output nodes yield linear output values.

3.3.6 Network Parameters. Following the recommendations of Orris and Feeser (1990A, 1990B), the parameters used in training the neural network metamodels are specifically listed as follows.

1. Error Tolerance. Training ceases when network training converges to within ±.05 error between the network estimate and actual value of the response variable sorties flown. The number is chosen, *prima facie*, on the grounds that a forecasting method that can consistently *predict* within this error tolerance is a relatively superior method.

It must be understood from the outset that the error tolerance used here is not equivalent to the Mean Squared Error (MSE) previously discussed in Hecht-Neilsen's theorem. Rather, it represents the ± error percentage of the output values derived by the network in comparison with those contained in the patterns of the training set.

2. Learning Rate and Smoothing Factor. As a staring point, the *Brainmaker Professional* (California Scientific Software, 1990) software default values of η=1.0 and μ=.9 are used for the learning rate and smoothing factor, respectively. These values were originally determined through empirical testing during software development, and have been successfully employed in a variety of applications (California Scientific Software Technical Support, 1991). In the event that convergence to within a pre-specified error tolerance does not occur, a plausible recommendation is to lower the learning rate to allow weight updates smaller in magnitude than those at higher learning rates (Lawrence and Lawrence, 1990:5-25), and retrain. Here, the alternate learning rate is initially set at η=.5 (in the

event convergence to within a pre-specified error level does not occur due to gradient descent error oscillation). As previously noted in *3.3.4*, the smoothing factor is left unaltered.   3. Minimum and Maximum Range Values. The modeling software used in this study automatically computes minimum and maximum values for all input and output nodes by calculating the values lying two standard deviations from the mean for each input and output vector in the training (fitting) set. By default, sample values above and below these computed values are "clipped", i.e., outlying sample values are taken to be equal to the applicable (computed) minimum or maximum, perhaps indicating a primary orientation toward pattern classification over generalized function mapping with respect to *Brainmaker Professional* software functionality. For this research, the sole concern is compromising (due to "clipping") the sorties flown output values (since all input values are equal to 0 or 1). The need to predict accurately across all the entire output range of sorties flown is critical. The default minimum and maximum values are manually reset to those that exist in the training sample response range, in the attempt to create a more representative test of predictive capability.

3.3.7 Architecture Determination Procedure. To facilitate comparative analysis (particularly across days), it appears desirable to create neural network metamodels possessing identical architectures. A fundamental concern in much of neural network research is determining the number of hidden layers required to represent a problem. A single hidden layer connecting 10 inputs to 1 output is chosen in the interest of maintaining a conservative research approach. Given this, an equally critical factor lies in determining the number of hidden nodes required for representing the problem under study. This number is determined as follows.

1. Using Hecht-Nielsen's backpropagation theorem (see *2.4.8.3*) as a starting point, each network is tested at 21 hidden nodes (this number equals twice the number of input nodes plus one). All networks are tested at this number for convergence to a (preliminary) selected error tolerance of ±.1.

2. On the advice of Dr. Kenneth Melendez (1991), this number (of hidden nodes) should be gradually reduced to prevent rote memorization of the training set and superfluous hidden nodes. Here, they are reduced one by one until the network under analysis will no longer attain the desired error tolerance (±.1). We note this attempt to find the minimum number of hidden nodes required for problem representation appears consistent with the observations of Hertz et al. (1991) (see *3.3.2.2* above), regarding model overspecification.

3. To ensure that no smaller number of hidden nodes will adequately represent the problem, the test is performed

in reverse, starting with 1 hidden node, incrementing this number by 1, and again stopping when the network in question converges to the error tolerance of ±.1.

4. The number of hidden nodes to be used for every metamodel in this study is established by using the largest number resulting from steps 2 and 3 above (the largest minimum number of hidden nodes required to reach the ±.1 error tolerance exhibited by any network metamodel of the 12 under analysis). In this way, all networks possess identical architectures, presumably rendering them amenable to exploratory analysis and comparison across days.

3.3.8 Network Training Procedure. The procedure used in training each neural network metamodel is listed as follows.

1. Each model's architecture is identically specified, as determined by the procedure discussed above.

2. Using the determined architecture specification, training is reinitialized (started anew) and the networks iteratively learn the pattern set (the training sample presented to allow weight modification) until convergence is gradually attained at ±.10, .08, .06, and, finally, .05 error tolerance levels. In the event that a given metamodel consistently fails to converge at one of the pre-determined tolerance levels after 10,000 iterations, the learning rate is set to .5 (η=.5) and training is reinitialized.

3. If convergence is not attained at the current error tolerance level after lowering the learning rate and allowing the network train for a maximum of 10,000 additional iterations at the current tolerance level, training is terminated. The rationale for training the networks beyond an encountered difficulty with convergence lies in the idea of providing an additional "margin" in which the network may learn more of the training set prior to (prediction) testing.

3.3.9 Summary. Two metamodel sets are formulated for analysis. The first contains 12 backpropagation neural networks modeling the first six days of the Attack and No-Attack Scenarios, while the second models precisely the same days and consists of fully specified regression models. The models are specified and trained/fitted as discussed above; goodness-of-fit and predictive accuracy are analyzed in terms of the following procedures.

3.4 Metamodel Evaluation

The following section delineates the three types of statistical measures used in this study for assessing model fit and predictive accuracy--measures of association, forecasting accuracy, and statistical significance.

3.4.1. Measures of Association (McClave and Benson, 1988:979-983). The degree of association between the control data provided by the simulation and that generated by the metamodel fits/forecasts (regression and network metamodels) is indicative of metamodel goodness-of-fit/ forecasting accuracy. This type of measure has been used previously in exploratory neural network research (Orris and Feeser, 1990A and 1990B). This study uses a nonparametric alternative, Spearman's Rank-Order Correlation Coefficient, r_s, which provides a measure of closeness or strength between actual data (simulated) values and a set of fitted or forecasted values for same. This method of measuring association is used as it cannot be assumed two given samples under analysis (actual *versus* forecast, paired by treatment) are drawn from the same underlying bivariate normal population: each simulation design point (treatment) possess its own distribution that has not been estimated *via* replication, hence Pearson's Product Moment Correlation Coefficient is inappropriate. Siegel and Castellan (1988:2-44) note that Spearman's r_s has an efficiency of about 91% in comparison with Pearson's r.

Spearman's rank correlation coefficient is given by McClave and Benson (1989:979- 983) as

$$r_s = SS_{uv} / (SS_{uu}SS_{vv})^{1/2}$$

where

$$SS_{uv} = \Sigma\ u_i v_i - (\Sigma\ u_i\ \Sigma\ v_i)/n$$

$$SS_{uu} = \Sigma\ u^2_i - (\Sigma\ u_i)^2/n$$

$$SS_{vv} = \Sigma\ v^2_i - (\Sigma\ v_i)^2/n$$

and

u_i = Rank of the ith measurement in sample 1

v_i = Rank of the ith measurement in sample 2

n = Number of pairs of measurements (number of in each sample.

The associated one-tailed test of statistical significance is given by

H_o: $\varrho = 0$

H_a: $\varrho > 0$ (the correlation is positive)

Test Statistic: r_s, the sample rank correlation.

Rejection Region: $r_s > r_s, r_\alpha$

3.4.2 <u>Measures of Forecast Accuracy</u>. Predictive accuracy is measured in terms of several statistics. Measures 1-6 listed below will be found in Makridakis et al. (1983:44-54); the reader is directed to that reference for full treatment, as well as to Armstrong (1978:319-333) for additional information. Each fundamentally analyzes error, where the forecasting error for a given observation is understood as $e_i = X_i - F_i$ (Makridakis et al., 1983:44), i.e., each error e_i is simply equal to the difference between the actual value X_i and the forecasted value F_i. (Throughout this section, the index *i* ranges from *1* to *n*, where *n* represents the total number of observations).

We focus upon Mean Error (ME) to indicate metamodel bias, Mean Absolute Error (also known as Mean Absolute Deviation) and Root Mean Squared Error (RMSE) measures to assess each model's relative forecasting accuracy, and the Adjusted Mean Absolute Percentage Error (AMAPE) for comparing metamodel forecasting accuracy across days. The AMAPE is selected due to its resilience to data measurement errors (Armstrong, 1978:322), a potential difficulty here in that each treatment response possesses its own (unknown) probability distribution. The measures are collectively defined as follows.

1. Mean Error (ME).

$$ME = \sum_i e_i / n$$

2. Sum of Squared Errors (SSE).

$$SSE = \sum_i e_i^2$$

3. Mean Squared Error (MSE).

$$MSE = SSE / n$$

4. Root Mean Squared Error.

$$MSE^{1/2}$$

5. Mean Absolute Error (MAE), (or Mean Absolute Deviation (MAD)).

$$MAE = \sum_i |e_i| / n$$

6. Standard Deviation of Errors (SDE).

$$SDE = (SSE / (n-1))^{1/2}$$

7. Adjusted Mean Absolute Percentage Error (AMAPE) (Armstrong, 1978:322).

$$AMAPE = \sum_i (|e_i| / (X_i+F_i)/2) / n$$

8. Error Range and Minimum Error. Maximum errors exist in both positive and negative directions, hence both

are recorded to reveal range. The minimum error is recorded as that value which is closest to 0 in terms of absolute value.

3.4.3 Measures of Statistical Significance. Two tests of significance are selected for determining whether forecasts are significantly different from simulated sorties flown values (the control groups) by way of differences paired by treatment. Prior to application of either, we analyze each set of sample differences for normality by way of the Wilk-Shapiro test. In the event that conditions of normality do not obtain for a given sample of differences, a non-parametric difference test is employed. The paired difference tests used in this research are listed as follows.

1. Two-Tailed Paired Difference Test of Hypothesis (Parametric) (McClave and Benson, 1988:454).

H_o: $(\mu_1 - \mu_2) = D_o$

H_a: $(\mu_1 - \mu_2) \neq D_o$

Test Statistic: $t = \dfrac{\bar{x}_D - D_o}{s_D / n_D^{1/2}}$

Rejection Region: $t < -t_{\alpha/2}$ or $t > t_{\alpha/2}$

where

$t_{\alpha/2}$ has $(n_D - 1)$ df (degrees of freedom).

2. Wilcoxon Signed Rank Test (Nonparametric) (McClave and Benson, 1988:959).

H_0: Two sampled populations have identical probability distributions.

H_a: The probability distribution for population A is shifted to the right or to the left of that for population B.

Test Statistic: T, the smaller of the positive and negative rank sums, T_+ and T_-.

Rejection Region: $T \leq T_0$.

3.5 <u>Limitations and Metamodel Relationships</u>

3.5.1 <u>Limitations</u>. Several factors limiting the potential predictive accuracy of both metamodel types must be noted as follows:

1. Sample size. For many applications, neural networks are trained on samples containing thousands of observations. Each network developed in this research is trained on a sample size of 128. Similar considerations regarding sample size adequacy (thus representativeness) apply for regression as well, but perhaps not to the same degree due to the robustness of the technique. It is noted that networks may not represent sufficiently the problem at a level when the number of weights (as determined by the number of hidden nodes) is greater than the number of observations contained in the sample and if sample sizes are too small in general.

2. Mapping dichotomous (binary) input vectors to a real-valued output. The observations comprising both the training and testing samples used in this research consist of an input vector whose 10 scalar components may take on

only values of 0 or 1, while the output value is an integer count of sorties flown (ranging roughly in value from 0 to an upper limit of 300). The possibility exists that real-valued inputs may yield a better problem representation from the networks under study than those used in this research; it is certain that larger samples are desirable. While not available for the network or regression metamodel fitting process in this research, such data may, (given sufficiently large samples), provide better representations from which to model, especially with respect to generalization (prediction) capabilities.

3. Neural Network Architecture. In essence, this problem is a hybrid of symbolic and real-valued (integer count) representation. As previously noted in the literature review (see *2.5.3*), real-valued function mappings often require two or more hidden layers, while symbolic mappings often predict most effectively *via* a single hidden layer. In the interest of conservatism, a single hidden layer representation is selected for this research, but the possibility remains that additional hidden layers may be appropriate for the problem under consideration, especially if used in conjunction with real-valued training sets. The difference essentially entails using a neural network to model the training set in terms of a hypercube or multi-dimensional surface, depending on whether one hidden layer or two (or more) are used, respectively.

4. Neural Network Learning Rates. Learning rates are critical (see Eqs (3.1) and (3.2)) in any network representation research. It should be stressed that the effect of different learning rates on a specified network topology constitutes an entire research area, not simply an empirical consideration. As previously justified in 3.3.5, the learning rate is fixed at 1, unless the network fails to converge (exhibits oscillatory behavior).

Thus, the "learning" equation used in this research stated above as Eq (3.2) reduces to

$$D_pW_{ij} = (1-\mu)(\delta_{pi}O_{pj}) + \mu(D_{p-1}W_{ij}) \qquad (3.5)$$

when the learning rate is set to 1 (η=1). It is noted that this may be far from optimal for an initial setting (depending on problem characteristics), and may require alteration during the training process--particulary if the network in training exhibits oscillation in attempting to settle to the pre-specified error tolerance.

3.5.2 Metamodel Relationships. The tests performed in this research entail comparing the performance of a linear method to a non-linear one. Hecht-Nielsen notes the following regarding the relationship between the two modeling techniques chosen for evaluation in this research:

> The manner in which mapping networks approximate functions can be thought of as a generalization of statistical regression analysis. In regression, the specific form of a function to be fitted is first chosen and then fitted according to some error criterion (such as the mean squared error). This

procedure is, at its core, based upon the *least mean of squared errors* (or simply *least squares*) technique for fitting a straight line to irregular data invented almost 200 years ago by Carl Gauss...A primary advantage of mapping networks over classical statistical regression analysis is that neural networks have more general functional forms than the well developed statistical methods can deal with. (1990:120)

In essence, the point of this research is to analyze and interpret the extent to which neural network models are or are not superior to multiple linear regression for predicting sorties flown for each scenario day under study, subject to the methodological constraints and recognized limitations noted above.

IV. Analysis and Findings

Overview

This chapter presents the analysis and findings of a comparison of regression and neural network metamodels fitted to simulated data previously generated by Diener (1989). A neural network metamodel and a fully specified linear regression counterpart are fitted to each daily sample of the twelve days under study (the first six days of Attack and No-Attack Scenarios), and subsequently compared on the basis of strength of correlation between fitted/ forecast and actual values, measures of forecasting error, and statistical significance between forecast and actual values.

The Chapter is organized in five Sections. In Section 1, we evaluate the network architecture determination and training process. Section 2 presents the measures of association used for assessing the degree of correspondence between fitted/forecast and actual (simulated) values (stated in *3.4.1*), and provides commentary on the significance of these measures. Section 3 reports the error statistics measuring relative and comparative forecasting accuracy of the two methods (*3.4.2*), while Section 4 presents the statistical results regarding the detection of significant differences between actual (simulated) and forecasted sorties flown response values (*3.4.3*). Finally, Section 5 provides a summary of the analyses conducted in this chapter and reviews the significant findings discovered. The thesis ends with general research conclusions and recommendations for future research, which jointly comprise Chapter 5.

4.1 Network Architecture and Training

4.1.1 Architecture Determination. Following the methodology of *3.3.7*, the minimum number of hidden nodes required for the set of 12 networks under study to reach convergence at the preliminary ±.1 error tolerance level is determined to be 14. Accordingly, each network to be trained to the final ±.05 error tolerance level possesses a 10-14-1 architecture, and the number of free parameters (weights) fitted by the gradient descent error minimization procedure operative during the training process is thus 169 ((10*14 + 14) + (14*1 + 1) = 169). It is noted that this number (169) exceeds the number of observations present in each daily training sample (128) (see *3.3.2.2*), but nevertheless appears to be required for *efficient* function mapping during network training (subsequent statistical analyses will reveal the extent to which the architecture is *effective* for the purpose of prediction). In contrast to this guideline, the number of observations present in each training sample (128) exceeds the total number of nodes in each network (at 14) by a factor of approximately 9, which meets the suggested practice that the number of observations exceed the number of nodes by a factor of at least 2-10 (*3.3.2.2*).

The number of hidden nodes required to meet the preliminary error tolerance level of ±.1 is illustrated in Figure 4.1. Beginning with one hidden node and successively

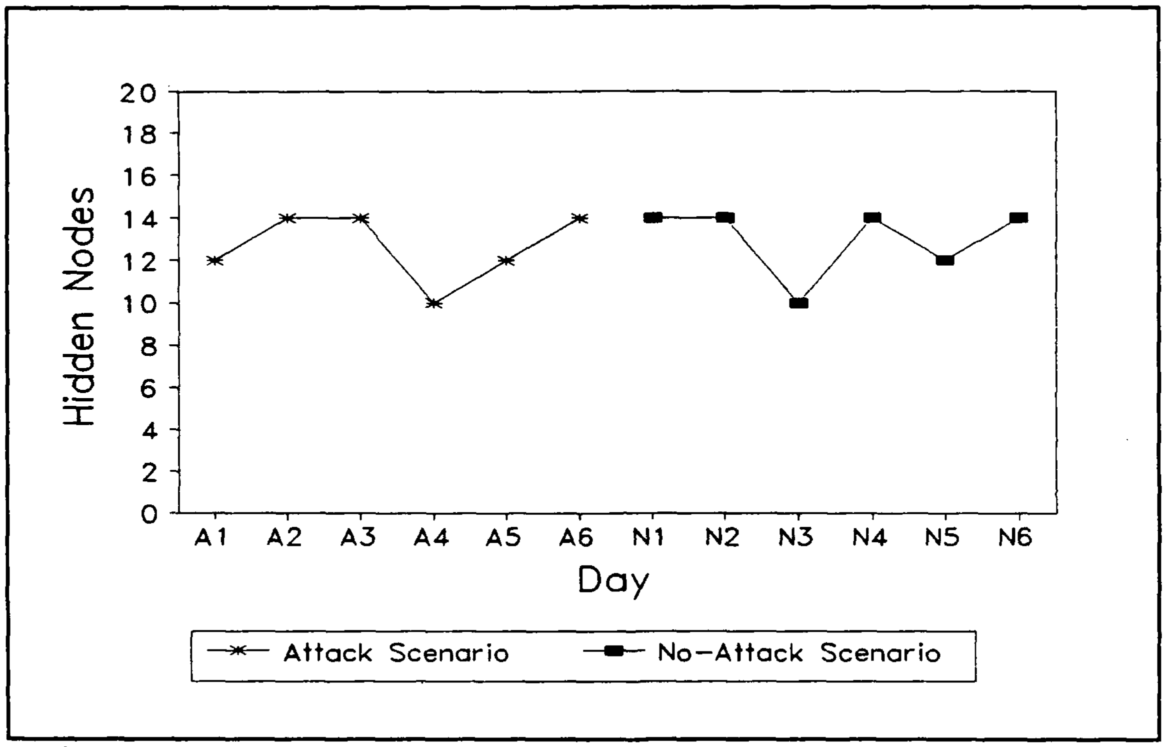


Figure 4.1 Hidden Nodes Required for ±.1 Error Tolerance

increasing their number by one (*3.3.7*), seven of the networks failed to reach the preliminary tolerance level until 14 hidden nodes were employed (at a learning rate of η=1.0 and smoothing factor of μ=.9). The process was repeated in *reverse*, beginning with twenty one hidden nodes (*3.3.7*) and reducing their number by one until the networks did *not* converge to the preliminary error tolerance level (±.1), with identical results (using the same learning and smoothing rates). The sole exception proved to be the network for Attack Scenario Day 3, which did not converge until the learning rate was lowered, noted in *4.1.2* below (14 hidden nodes were required for this network as well).

While five of twelve networks converged at lower hidden node levels (fewer than 14), the initial design decision to configure all networks with identical architectures to facilitate their comparison with regression counterparts requires the use of 14 (hidden nodes) for each:

1. All networks are constructed with identical architectures in the attempt to ensure they are fully-specified and not optimized for fitting or forecasting *via* pruning, training with noise, etc., to provide a fair comparison with their regression counterparts.

2. While five of the network metamodels met the *preliminary* error tolerance at hidden node levels below 14, each failed to converge to the established *fitting* error tolerance of ±.05 in additional testing. When specified at 14 hidden nodes, all networks converged to the fitting tolerance.

Thus, in accordance with previously established methodology and the above rationale, all networks are specified at a 10-14-1 architecture prior to training.

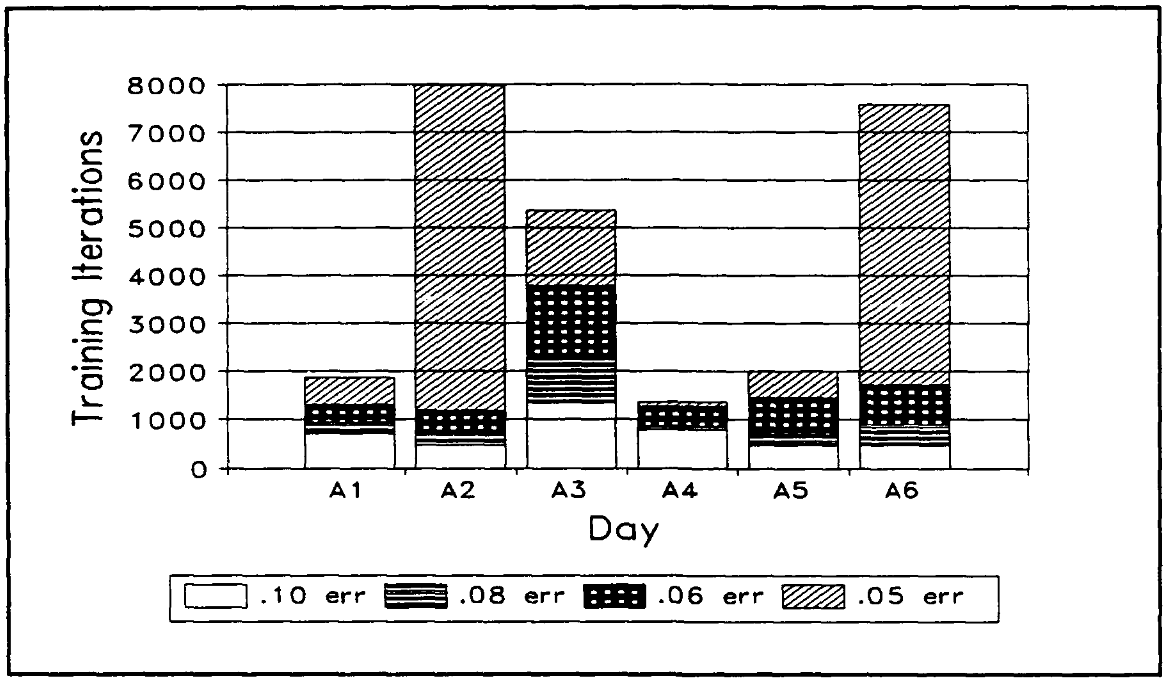
4.1.2 <u>Network Training.</u> All networks were successfully trained to within the selected error tolerance of ±.05; the weight matrices derived during training are listed for each daily network metamodel in Appendix A. In all cases except Attack Scenario Day 3, a learning rate of η=1.0 and smoothing factor of μ=.9 brought about convergence to the desired error tolerance. A3 (henceforth, days are preceded by *A* or *N* to indicate Scenario) converged only after reinitializing training with a learning rate set at η=.5. Table 4.1 lists training iteration counts and corresponding elapsed training times for each neural network metamodel. In this table, cumulative iterations are displayed under the associated error tolerance level for each day, while the last column lists the total elapsed clock time (in minutes) required by each network to reach the final ±.05 error tolerance level. Figure 4.2 graphically depicts the cumulative iterations required to reach the error tolerances listed in Table 4.1, for Attack and No-Attack Scenarios, respectively.

In general, Attack Scenario network training iterations seem largest at the ±.05 tolerance level (see the .05 boxes in Figure 4.2) in comparison to the No-Attack Scenario, which appears to be concentrated at both .06 and .05 tolerance levels. Further, the general level of effort required to achieve network error tolerances seems more homogeneous across days for No-Attack nets: the mean and standard deviation of training iterations are 4,379 and 2,995.91 for the Attack metamodels, and 2,550.5 and 1,070.84 for their No-Attack counterparts. While clearly interpretive, the fact that Attack Scenario day training sets exhibit more variation in response values than No-Attack samples appears to be evidenced by the larger variance in iterations required for Attack metamodel convergence.

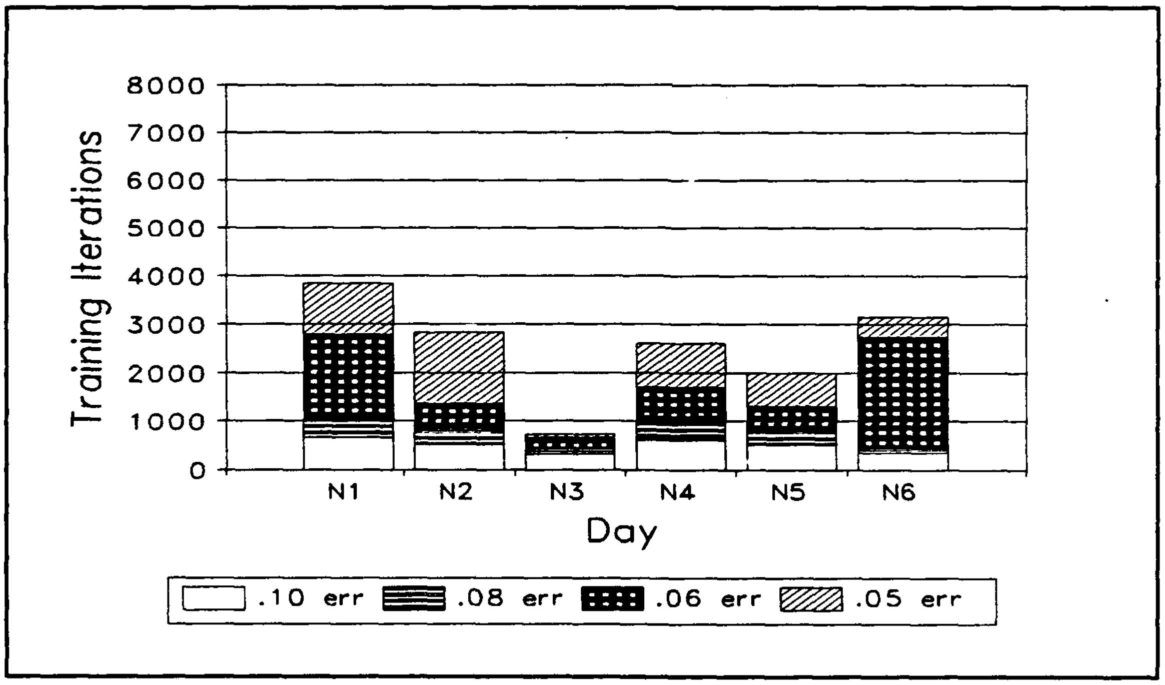
Table 4.1

Training Iterations and Elapsed Clock Time

Day	Training Error Tolerance Levels and Associated Cumulative Iterations				Minutes to .05 Error Tolerance
----	.10	.08	.06	.05	--------
A1	704	938	1315	1886	62.87
A2	477	728	1207	7991	266.37
A3	1356	2306	3809	5388	179.60
A4	787	861	1299	1390	46.33
A5	490	711	1493	2046	68.20
A6	476	907	1729	7576	252.53
N1	666	1067	2816	3872	129.07
N2	519	814	1375	2848	94.93
N3	326	441	678	753	25.10
N4	608	1099	1711	2624	87.47
N5	515	825	1334	2030	67.67
N6	359	473	2766	3176	105.87



Attack Scenario



No-Attack Scenario

Figure 4.2 Training Iterations

4.1.3 Overtraining. A general concern during backpropagation network training is *overtraining*--networks may fit the training set too closely to allow for adequate prediction. Hecht-Nielsen notes:

> An unexpected and peculiar phenomenon found in some feature-based mapping networks (notably, backpropagation) is the problem of *overtraining*....The source of the problem seems to be a tendency of some networks to start out by implementing a very "flat" approximating function....
>
> As training of one of these peculiar mapping networks progresses, it seems to be the case that the initially flat surface defined by the functional form of the network begins to "crinkle" and develop undulations -- as it must if it is to better fit the training set examples. However, as this process continues, if the set of training examples are [sic] shown repeatedly to the network many, many times, then the surface becomes even more crinkled and convoluted in its attempt to fit this fixed set of points. In behavioral terms, all it "cares about" is fitting these points. The ability to interpolate well between them is unimportant. (1990: 116-117)

Clearly, error tolerance settings determine *goodness-of-fit* for the networks in this study--they essentially specify the degree of "closeness" the network metamodel computed response values must come to actuals comprising the training sets. Interpreting Hecht-Nielsen's observation in terms of error tolerances, it appears that the smaller the tolerance, the greater the possibility that network may predict poorly, *ceritus paribus*, as more and more training iterations are required to minimize the error between (network) computed and actual values. Thus, there is no guarantee that the error displayed in forecasting unseen cases will be similar in magnitude to that derived during training, although, given sufficiently large and representative training and testing sets, this would be expected. As a first examination of how well the networks predict the testing sets in comparison to how they fit the training sets, we turn to an examination of correlation for both fitted and forecasted values.

4.2 Metamodel Correlations

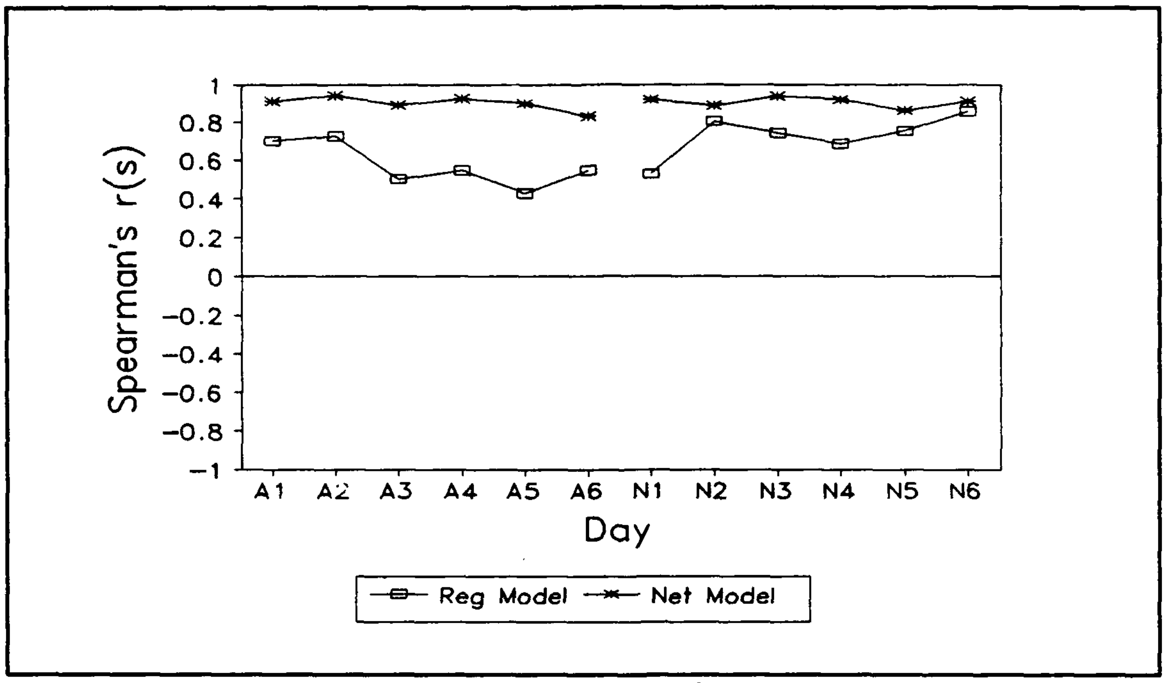
Table 4.2 presents Spearman's Correlation Coefficients for each network and regression metamodel for both fitting (training) and forecasting (prediction) testing, while Figure 4.3 plots same. Note that the statistics and graphics are provided for networks trained to the final error tolerance of ±.05, and than the graphs are broken into separate Attack and No-Attack Scenarios. The correlations tabulated and plotted measure the closeness between actual and fitted estimates of training (fitting) set values in the case of model *fits*, and the closeness of actual and forecast testing set values in the case of model *forecasts*. Throughout this chapter, when the terms *fit* and *forecast* are used, they retain these connotations.

4.2.1 Fit and Forecast Correlations. As would be expected, the non-linear networks exhibit consistently larger positive correlations that the linear regression metamodels with respect to model fitting, as reflected in the first graph of Figure 4.3. With respect to forecasting, the trend is notably reversed--network correlations show significant degradation with respect to the prediction of unseen cases, particularly for Attack Scenario networks.

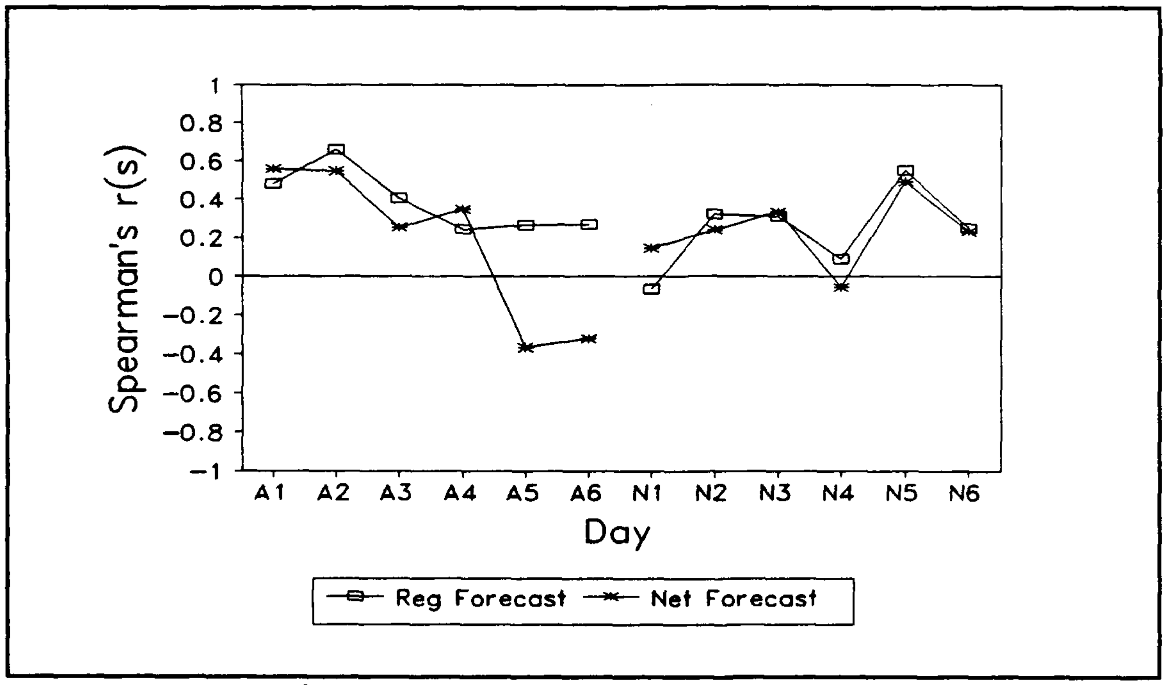
Table 4.2

Fit and Forecast Rank Correlation Coefficients

Day	Reg Fit	Net Fit	Reg Forecast	Net Forecast	Reg α-Level	Net α-Level
A1	.7017	.9117	.4808	.5600	.0250	.0100
A2	.7281	.9407	.6614	.5489	.0025	.0050
A3	.5027	.8907	.4030	.2526	.0500	.2500
A4	.5480	.9228	.2442	.3474	.2500	.1000
A5	.4309	.8986	.2640	-.3678	.2500	----
A6	.5487	.8312	.2716	-.3236	.2500	----
N1	.5308	.9205	-.0638	.1464	----	----
N2	.8062	.8887	.3214	.2401	.1000	.2500
N3	.7450	.9355	.3124	.3352	.1000	.1000
N4	.6867	.9203	.0904	-.0565	----	----
N5	.7549	.8621	.5535	.4910	.0100	.0250
N6	.8590	.9127	.2503	.2321	.2500	.2500



Regression and Network Fitting Correlations



Regression and Network Forecast Correlations

Figure 4.3 Regression and Network Correlations

In the majority of cases, regression metamodels exhibit higher positive forecast correlations than their network counterparts. The two rightmost columns of Table 4.2 show alpha significance levels for the *forecast* correlations of both modeling methods; the superiority of the regression metamodels over the networks is amplified by the α significance levels (one-tailed test, $r_s>0$) listed, although neither metamodel type exhibits significantly high correlations across all days within either Scenario. The exceptions to regression forecast correlation superiority are days A1, A4, N1, and N3, where the network metamodels fare better.

The network metamodels for days A5, A6 and N4 exhibit negative forecast correlations, indicating an inverse relationship (with respect to direction) between actual and forecast values; in each of these cases, the correlations are not statistically significant. The regression metamodel forecast correlations for days A5 and A6 are somewhat better; for days N1 and N4, however, neither metamodeling method yields a significant α level. In general, correlations are statistically significant for regression forecasts (with the exceptions of day N1 and N4); networks fare poorly in this regard, with no significant correlations occurring for days A5, A6, N1, and N4 (1/3 of the total network metamodel set).

A particularly interesting phenomenon is noted in Figure 4.3 for No-Attack Scenario forecasts--network metamodels appear very similar, in day to day correlational trend levels, to their regression counterparts. While less significant in terms of α level than regression, the trends in this figure suggest that network metamodels closely approach the correlational values exhibited by regression in the No-Attack Scenario, but fare poorly for Attack days (notably A5 and A6). A plausible explanation for this phenomena is that the networks have more difficulty in predicting Attack Scenario response values due to response variability inherent in Attack training and testing samples. Regression metamodels, on the other hand, appear more robust in dealing with response variation.

4.2.2 Forecast Correlations and Training Iterations. Forecast correlations and training iterations are compared in Figure 4.4. In general, the number of iterations required for convergence to $\pm.05$ tolerance during training appears quite dissimilar for corresponding days *between* Scenarios, and no convincing pattern appears for successive days *within* Scenarios. Further, no pronounced relationship between *forecasting* correlation and number of iterations required during *training* is indicated, although for 4 of the 12 metamodels, correlations are relatively high at low training iteration levels.

As noted, little if any indication of trend in iteration levels between successive days within a *given* Scenario seems present. While the presence of one would not, by necessity, imply network metamodel recognition of autocorrelation, it is plausible that modeling correlations across days (see *1.3.3* and *2.2.4*) could be beneficial, in that more information would be provided the networks. Here, it appears that a given daily network metamodel knows nothing of its predecessor or successor, which is consistent with the cross-sectional metamodeling approach adopted from the outset of the study. While clustering of relatively high forecast r_s values at lower training iteration levels has been noted, there is no way to examine an *overtraining hypothesis* that relatively high r_s values generally occur at lower iteration levels without a basis of comparison. A possible means of exploring this possibility--comparing the strength of forecast correlations of daily metamodels trained to successive error tolerance levels--is presented next.

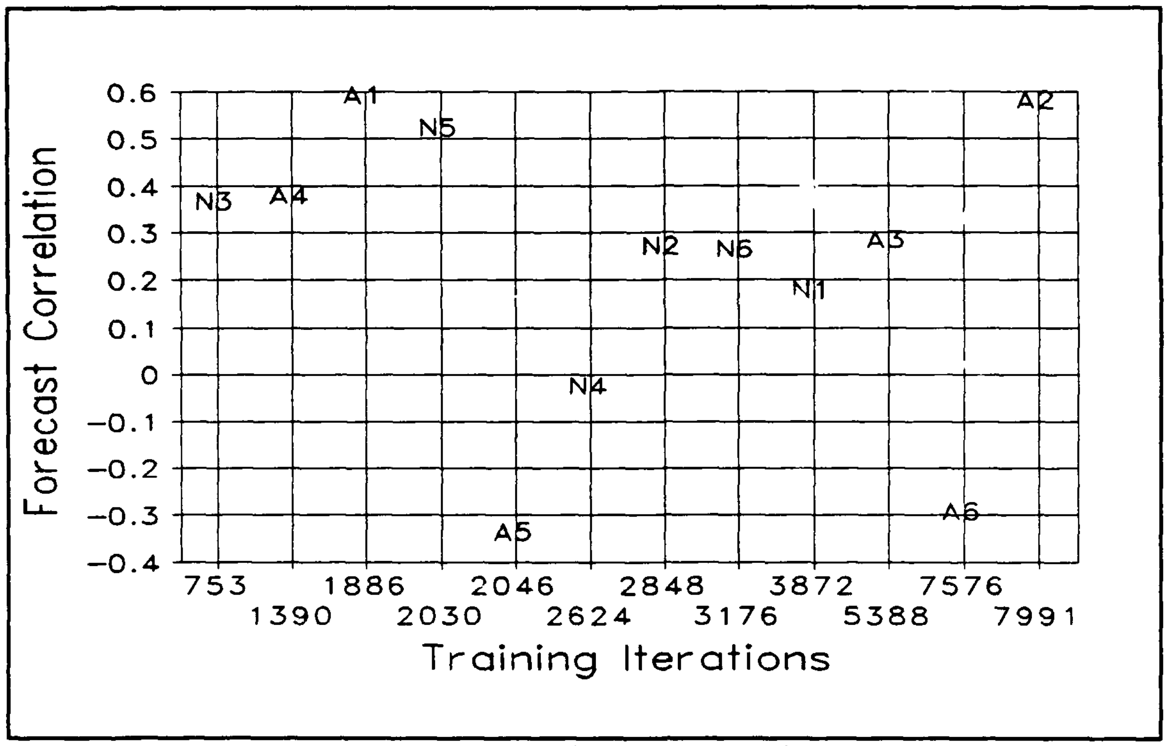
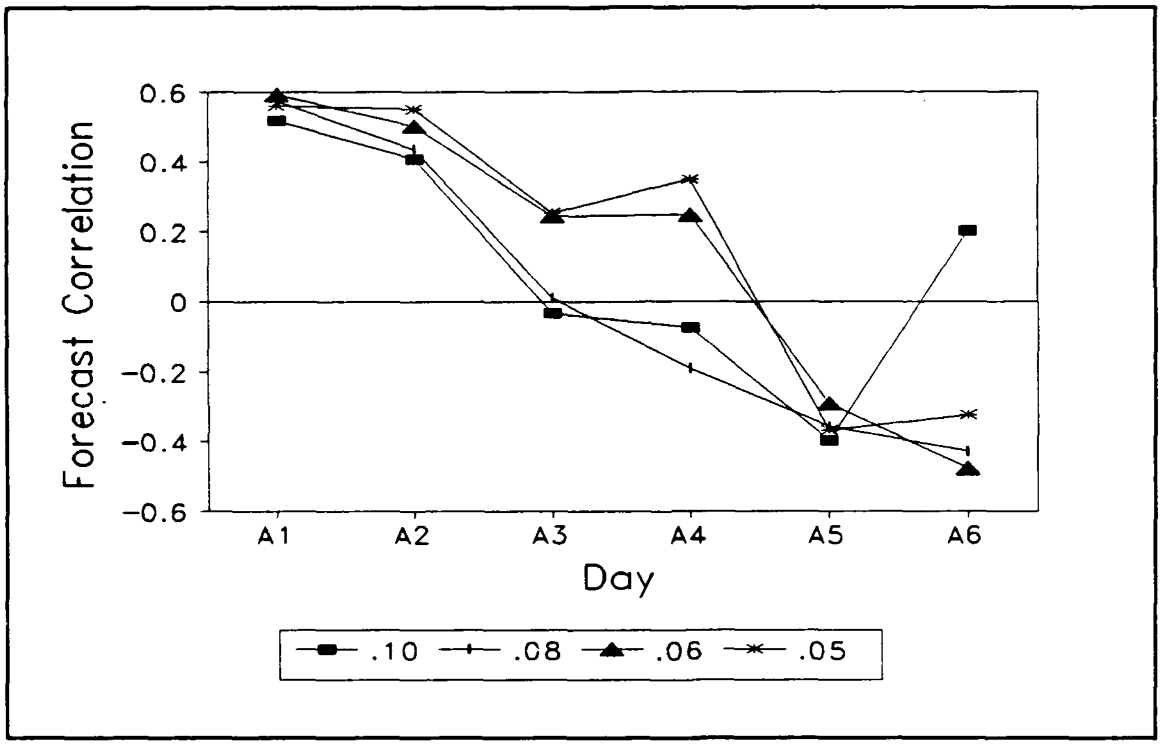


Figure 4.4 Forecast Correlations *vs* Training Iterations

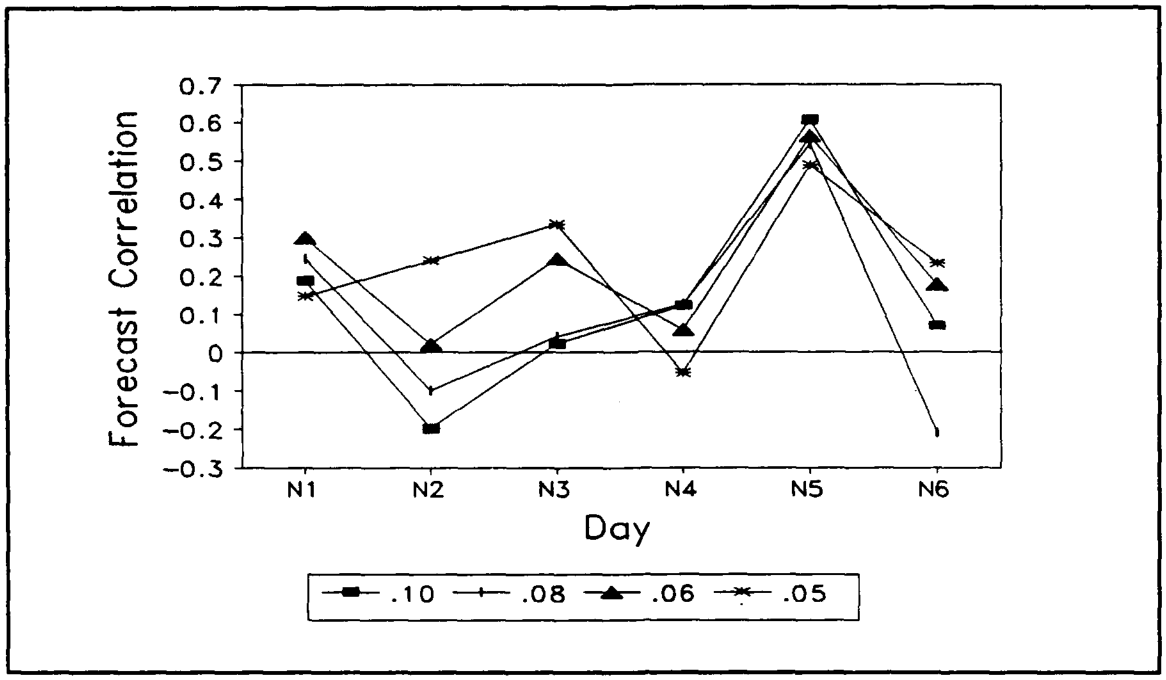
4.2.3 Forecast Correlations and Error Tolerance Levels. Spearman's Rank Correlation Coefficient is used to measure strength of association only--it is not a direct measure of forecasting accuracy. It does, however, provide an indication of the degree to which forecast values follow a linear direction or association with actuals comprising each independent test set. Figure 4.5 plots Spearman's r_s for forecast versus actual test set values for each day at .10, .08, .06 and .05 error tolerance levels, for each Scenario (networks saved at each tolerance level are used to generate their respective plot lines).

For both Scenarios, about 1/2 of the models reach their highest forecast correlation level (r_s) with networks trained to the .05 error tolerance level. While beyond the scope of this work, a study of correlations and predictive accuracy exhibited by networks trained at various error tolerance levels could test whether models fitted at higher tolerances provide better forecasts than those fitted at lower ones. Figure 4.5 indicates this with respect to forecast correlations, for this study.



Attack Correlations and Error Tolerance

Figure 4.5 Correlations and Error Tolerance



No-Attack Correlations and Error Tolerance

Figure 4.5 Cont'd Correlations and Error Tolerance

4.2.4 Summary. In this section, the topics of model architecture, training times, and metamodel fit/forecast correlations (with simulated actuals) were discussed. Network metamodels are found superior in fitting but inferior in forecasting in comparison with their regression counterparts. Further, 4 of 12 network metamodels did not possess statistically significant r_s correlations; only two regression metamodels failed to be statistically significant. No network metamodel attained a forecast correlation higher than .56; regression correlations were generally higher, the highest being .66 for day A2.

There is indirect evidence to suggest that networks trained at higher error tolerances may fare better in prediction than those fitted at lower ones. Networks also appear to approximate their regression counterparts (in forecast correlation level) more closely for the No-Attack than the Attack Scenario, which is partially attributable to the networks' seemingly less than optimal ability to account for response variation.

4.3 Forecast Error

This section presents measures of forecast bias, relative and comparative accuracy, dispersion, and other statistics, for both metamodel types. In practical terms, neither technique (regression and network) exhibits forecast error levels sufficiently low for real-world applications. In the following analysis, the chief points of interest center upon discovering the conditions in which the behavior of the metamodeling types is similar or dissimilar.

4.3.1 Forecast Bias and Relative Accuracy. Table 4.3 lists forecast bias and relative error measures; the three principal statistics of interest in the table are plotted in Figure 4.6, which presents forecast Mean Error (ME), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), in turn.

Both regression and network metamodels generally agree in sign (±) for ME (systemic bias); they disagree only for day A6. Also, both metamodel types are similar in trend (of magnitude) across days within Scenario, especially for No-Attack Scenario days. Networks metamodels exhibit smaller

MEs for days A2, A6 and N3, while regression metamodels fare better for the remainder.

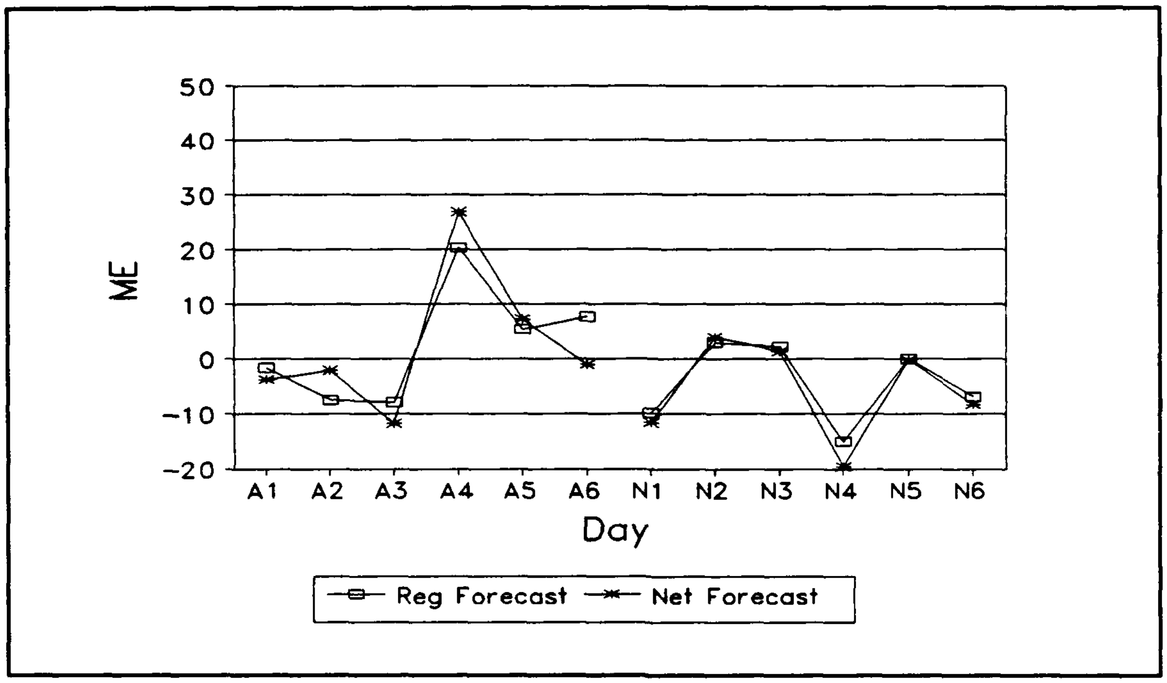
The reader is reminded to view Mean Error primarily in terms of bias; typical forecast error is measured more accurately by Mean Absolute Error (MAE), and by Root Mean Squared Error (RMSE). For both MAE and RMSE, Figure 4.6 indicates the same pattern in metamodel error for the No-Attack Scenario as that exhibited in the ME plot. The sole case of network superiority here (MAE/RMSE) is the MAE for day N3. For the No-Attack Scenario, however, the differences in MAE and RMSE between model types are much smaller than those observed for the Attack Scenario.

4.3.2 Forecast Error Range and Comparative Accuracy. Range statistics and the Adjusted Mean Absolute Percentage Errors (AMAPEs) are tabulated for each daily metamodel in Table 4.4. Figure 4.7 plots forecast error ranges for Regression and Network metamodels, respectively. In these plots, tick marks specify the Standard Deviation of Error (SDE) and the minimum absolute error for each metamodeling type's forecasts. Figure 4.8 plots the comparative AMAPEs; as a percentage error measure, AMAPEs allow comparisons of accuracy across days.

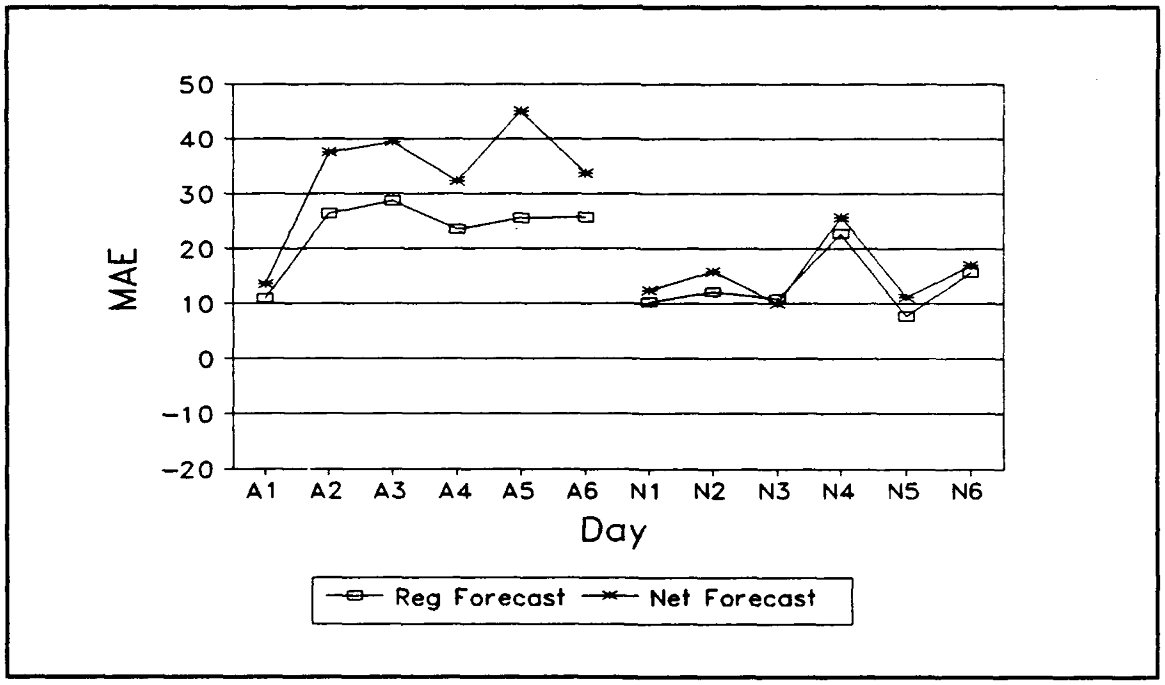
Table 4.3

Forecast Errors--Relative Accuracy Measures

Day	Reg or Net	SSE	MSE	ME	MAE	RMSE
A1	R	4,431.0682	221.5534	-1.546	10.967	14.885
A1	N	6,448.7772	322.4389	-3.704	13.590	17.957
A2	R	20,696.9709	1,034.8485	-7.406	26.445	32.169
A2	N	39,187.2889	1,959.3644	-1.991	37.673	44.265
A3	R	31,247.6316	1,562.3816	-7.891	28.714	39.527
A3	N	53,559.8143	2,677.9907	-11.721	39.425	51.749
A4	R	16,380.7183	819.0359	20.231	23.657	28.619
A4	N	33,735.0478	1,686.7524	26.855	32.369	41.070
A5	R	20,641.3357	1,032.0668	5.477	25.532	32.126
A5	N	57,081.4717	2,854.0736	7.211	45.004	53.424
A6	R	19,917.5500	995.8775	7.645	25.675	31.557
A6	N	38,537.2262	1,926.8613	-1.051	33.794	43.896
N1	R	2,729.9200	136.4960	-9.880	10.220	11.987
N1	N	4,019.2110	200.9606	-11.555	12.362	14.176
N2	R	4,870.7900	243.5395	2.905	12.125	15.606
N2	N	7,533.5897	376.6795	3.978	15.838	19.408
N3	R	3,380.6000	169.0300	2.240	10.850	13.001
N3	N	3,723.7669	186.1883	1.435	10.030	13.645
N4	R	17,352.4100	867.6205	-15.115	22.735	29.455
N4	N	23,032.8414	1,151.6421	-19.576	25.787	33.936
N5	R	2,869.3100	143.4655	-0.055	7.725	11.978
N5	N	4,534.7438	226.7372	-0.227	11.098	15.058
N6	R	8,095.3400	404.7670	-6.940	15.760	20.119
N6	N	8,756.4457	437.8223	-8.342	16.976	20.924

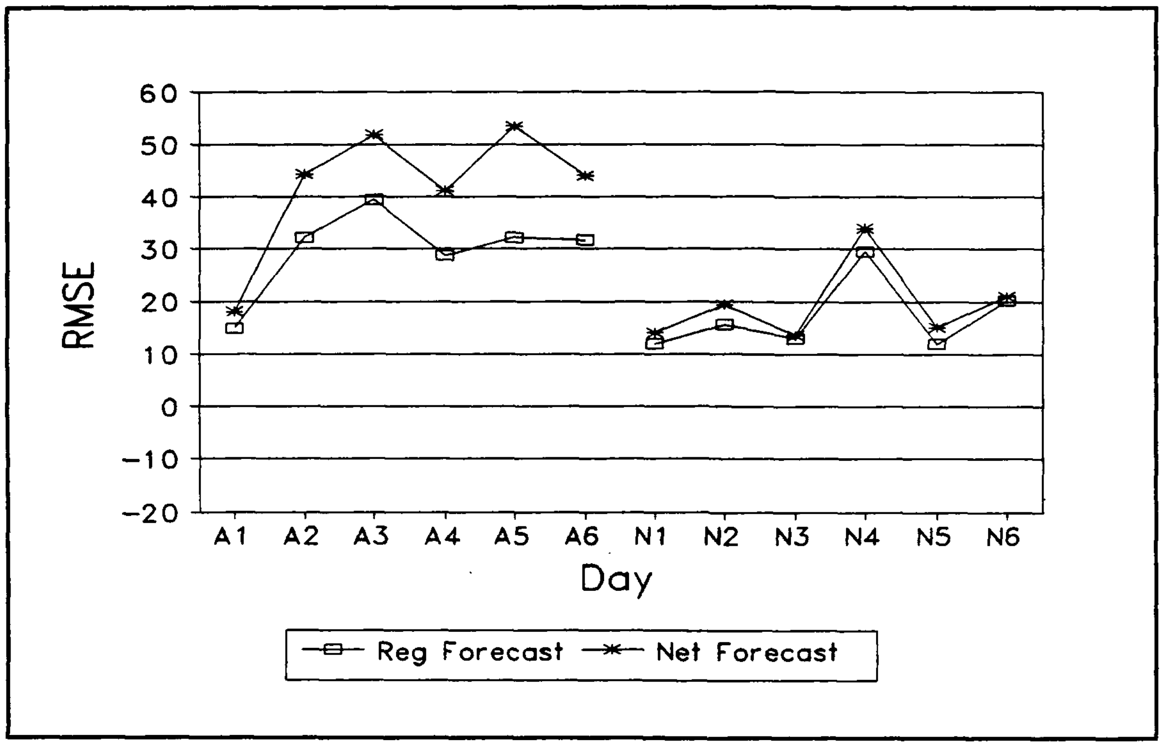


Forecast Mean Error



Forecast Mean Absolute Error

Figure 4.6 Forecast Errors



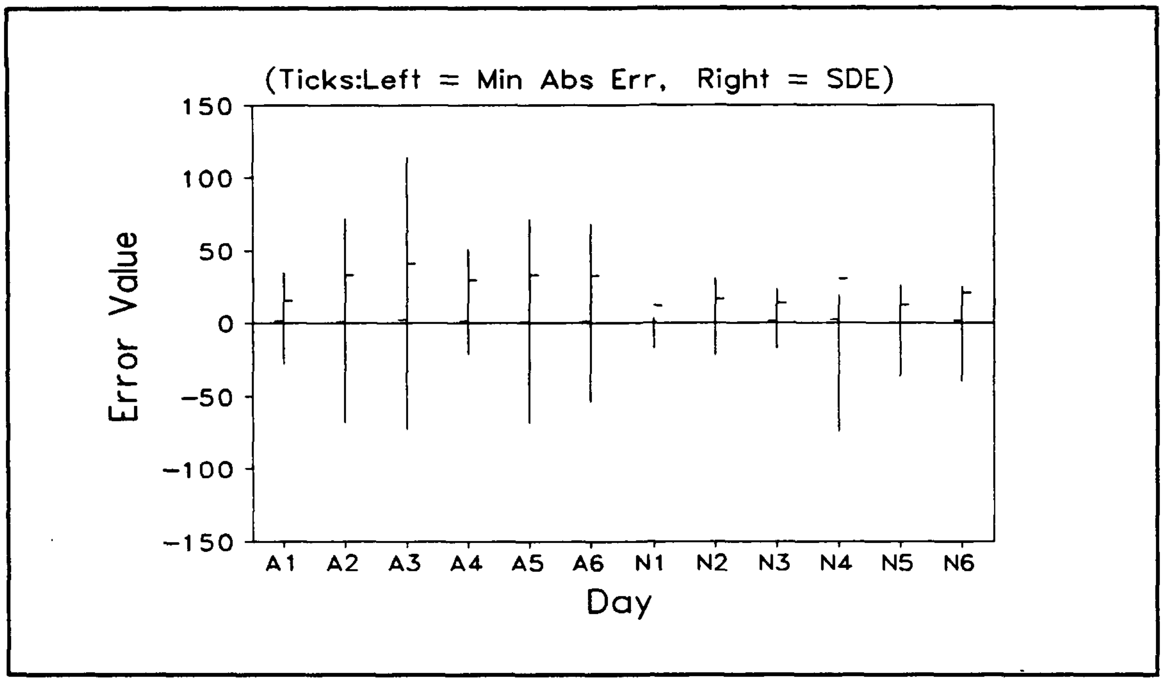
Forecast Root Mean Squared Error

Figure 4.6 Cont'd Forecast Errors

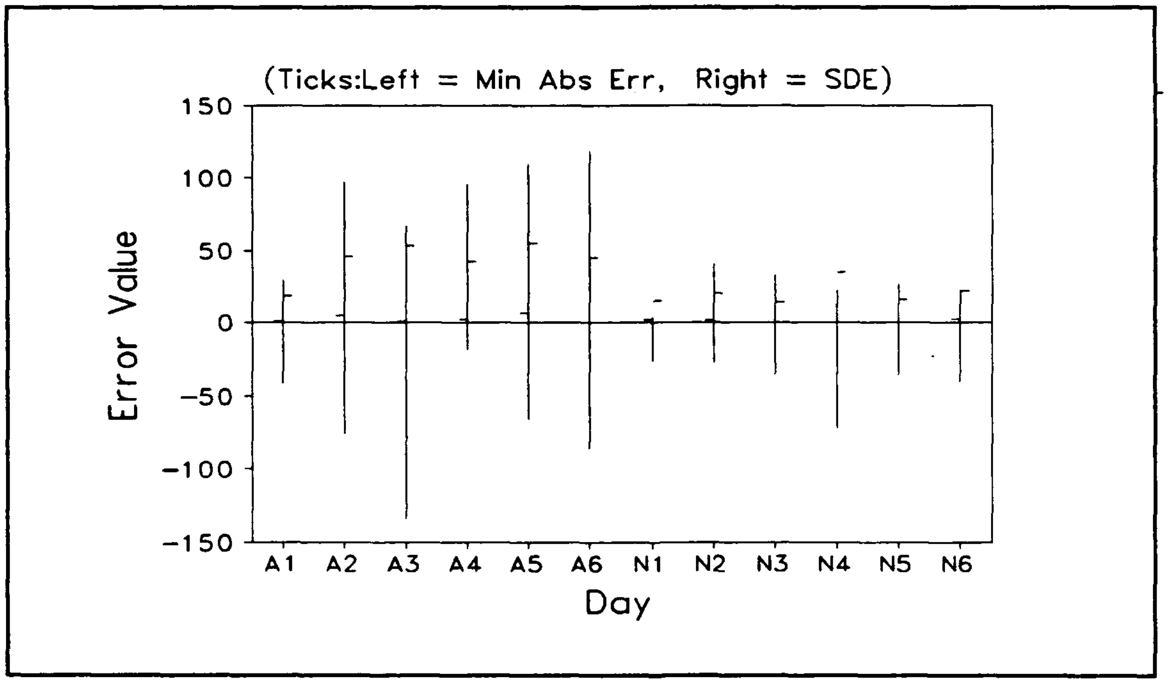
Table 4.4

Forecast Errors--Range and Comparative Accuracy Measures

Day	Reg or Net	Low Error	High Error	Abs Min Error	SDE	AMAPE
A1	R	-28.00	34.500	1.031	15.271	11.460
A1	N	-41.860	29.100	0.411	18.423	14.173
A2	R	-68.300	71.750	0.656	33.005	63.573
A2	N	-76.600	96.977	5.115	45.415	79.420
A3	R	-72.875	114.25	2.400	40.554	30.754
A3	N	-134.750	66.836	0.557	53.094	39.264
A4	R	-21.400	50.563	0.813	29.362	24.931
A4	N	-19.070	95.495	2.200	42.137	39.900
A5	R	-68.400	71.336	0.300	32.960	25.129
A5	N	-66.300	109.710	6.270	54.811	43.195
A6	R	-53.800	68.400	0.400	32.377	15.727
A6	N	-86.740	118.219	0.090	45.036	21.046
N1	R	-18.100	3.200	0.200	11.987	3.945
N1	N	-25.740	3.990	1.410	14.544	4.731
N2	R	-22.800	30.700	0.100	16.012	5.587
N2	N	-27.430	40.800	0.970	19.912	7.440
N3	R	-17.600	23.300	0.900	13.339	5.143
N3	N	-35.650	33.200	0.050	14.000	4.696
N4	R	-74.700	18.900	2.200	30.221	12.196
N4	N	-72.320	22.270	0.190	34.817	13.550
N5	R	-36.600	26.000	0.100	12.289	3.918
N5	N	-35.720	26.590	0.110	15.449	5.760
N6	R	-40.200	24.600	1.600	20.642	8.494
N6	N	-40.500	22.230	2.410	21.468	9.263



Regression Forecast



Network Forecast

Figure 4.7 Error Range and Dispersion

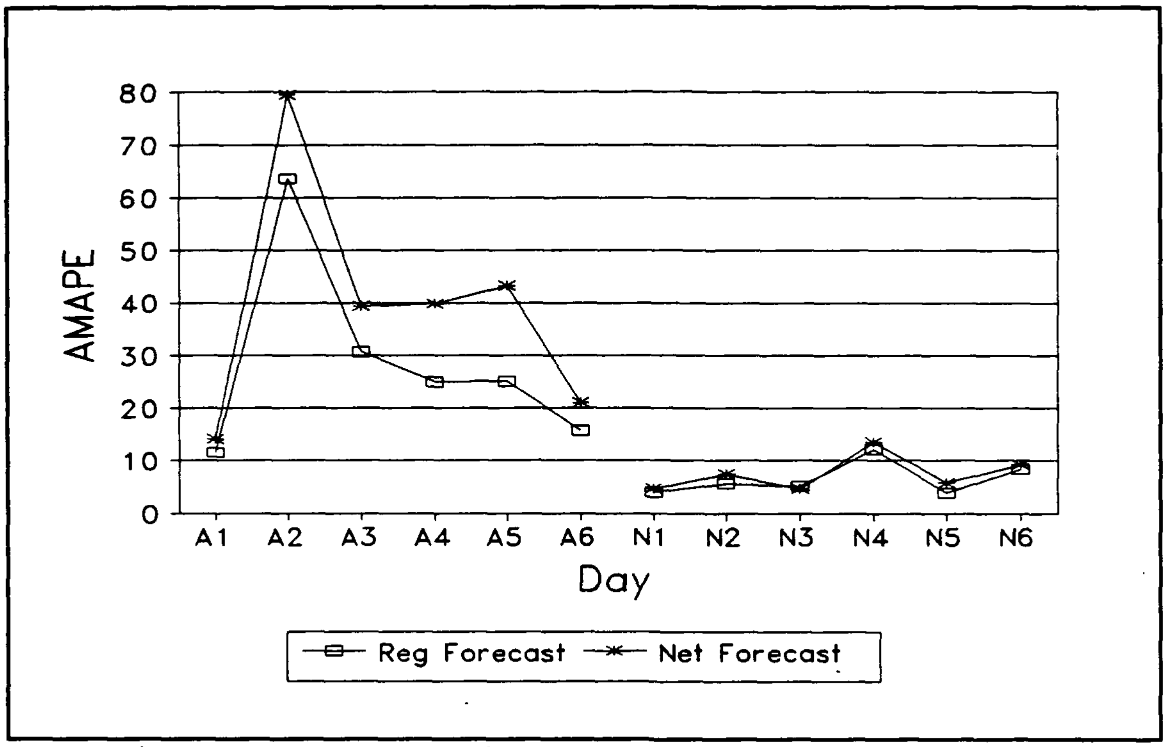


Figure 4.8 Regression and Network Forecast AMAPE

In general, regression metamodels exhibit much smaller error ranges in comparison to their network counterparts, and their Standard Deviations of Error (SDEs) are both smaller and more consistent from day to day. With respect to AMAPE, regression metamodels exhibit lower AMAPEs in all cases except day N3, indicating superior forecasting performance over the network metamodels.

Of particular interest is the similarity of pattern between regression and network metamodeling RMSEs and AMAPEs as illustrated in Figures 4.5 and 4.7. Network metamodel error trends parallel those of regression, albeit at generally higher levels; this is especially true for AMAPE. The difference between methods is relatively small for No-

Attack Scenario Days, although similarity in trend for the Attack Scenario is clear. As in the results for forecast correlations (see *4.2.1*), the implication is that the network metamodels appear to be *approximating* the performance of their regression counterparts. To repeat: the magnitude of network error for Attack days appears to be attributable to the network's seeming inability to deal with a high degree of response variation.

4.3.3 Summary. In this Section, the topics of forecast bias, relative and comparative accuracy, error range and dispersion, and error trend were discussed. Both metamodeling techniques generally agree with respect to bias (ME), and with the exception of the MAE for day N3, regression metamodels were consistently better in terms of relative accuracy (MAE and RMSE). Further, error ranges and SDEs were consistently smaller for regression, as were AMAPEs, again with the exception of day N3.

Neither model predicted test set data sufficiently well to be considered for real-world implementation. What is interesting is the degree to which network metamodels parallel the performance of their regression counterparts, albeit at higher error levels. In particular, networks closely approximate regression performance for No-Attack Scenario days; their relatively poor performance in modeling Attack days appears to be a function of inherent difficulty in dealing with response variation.

4.4 Statistical Significance

In this section, the differences between forecast and actual values are examined for statistical significance, for each metamodeling type. As a necessary prerequisite to the application of a paired difference experiment, the samples of differences (between each method and actuals) is examined for approximate normality *via* the Wilk-Shapiro (W-S) test. The results of the W-S test are listed in Table 4.5; all difference samples appear relatively normally distributed.

Tables 4.6 and 4.7 list t and p-values for the paired difference experiment, while Figures 4.8 and 4.9 plot same, respectively. The hypothesis under investigation is whether or not forecast values are significantly different from sample actuals, in accordance with the following test specification:

H_o: $(\mu_1 - \mu_2) = D_o$

H_a: $(\mu_1 - \mu_2) \neq D_o$

Test Statistic: $$t = \frac{\bar{x}_D - D_o}{s_D / n_D^{1/2}}$$

Rejection Region: $t < -t_{\alpha/2}$ or $t > t_{\alpha/2}$

where

$t_{\alpha/2}$ has $(n_D - 1)$ df (degrees of freedom).

For the testing sample size of $n=20$, the critical t value at the $\alpha=.05$ is 1.729 (McClave and Benson, 1988:1197).

Table 4.5

Wilk-Shapiro Tests

Day	Act-Reg	Act-Net
A1	0.8878	0.9248
A2	0.9468	0.8643
A3	0.8809	0.8493
A4	0.9696	0.9554
A5	0.9431	0.9279
A6	0.9736	0.9046
N1	0.9628	0.9140
N2	0.9698	0.8849
N3	0.9654	0.9488
N4	0.9365	0.8726
N5	0.8686	0.9256
N6	0.9647	0.9540

The computed t-values listed in Table 4.6 and plotted in Figure 4.9 indicate that both methods' forecasts for days A4, N1, N4 and N6 are significantly different than sample actuals; these results are amplified by the small p-values exhibited in Table 4.7 and plotted in Figure 4.10.

Table 4.6

t-Test t-values for Actual vs Forecast Values

	Regression			Network		
Day	Mean	StdErr	t	Mean	StdErr	t
A1	-1.546	3.3960	-0.460	-3.7040	4.0310	-0.920
A2	-7.406	7.1820	-1.030	-1.9910	10.1400	-0.200
A3	-7.891	8.8860	-0.890	-11.7200	11.5600	-1.010
A4	20.230	4.6440	4.360	26.8500	7.1290	3.770
A5	14.430	5.8280	2.480	7.2110	12.1400	0.590
A6	7.645	7.0240	1.090	-1.0510	10.0700	-0.100
N1	-9.880	1.4310	-6.910	-11.5500	1.8840	-6.130
N2	2.905	3.5180	0.830	3.9780	4.3580	0.910
N3	2.240	2.9380	0.760	1.4340	3.1130	0.460
N4	-15.120	5.8000	-2.610	-19.5800	6.3600	-3.080
N5	-0.0550	2.7480	-0.020	-0.2270	3.4540	-0.070
N6	-6.940	4.3320	-1.600	-8.3410	4.4020	-1.890

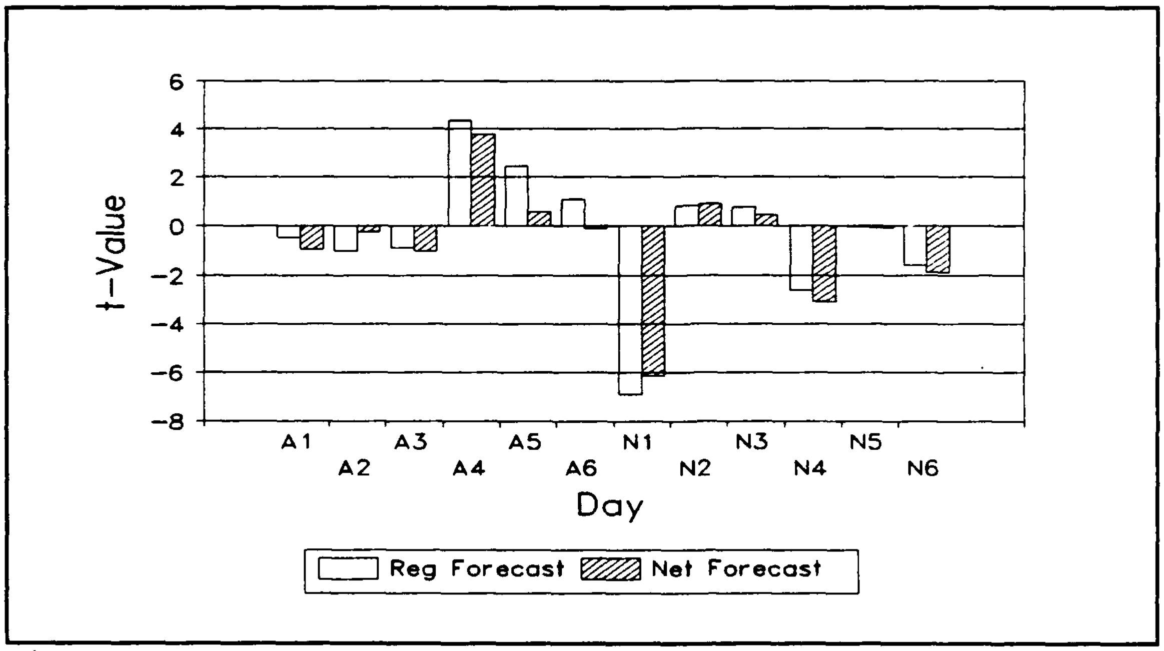


Figure 4.9 Paired Difference t-Values

Table 4.7

p-values for Actual vs Forecast Values

	Reg	Net
Day	p-value	p-value
A1	.6452	.3697
A2	.3154	.8465
A3	.3856	.3235
A4	.0003	.0013
A5	.0228	.5597
A6	.2900	.9179
N1	.0000	.0000
N2	.4191	.3728
N3	.4552	.6502
N4	.0174	.0062
N5	.9842	.9483
N6	.1257	.0734

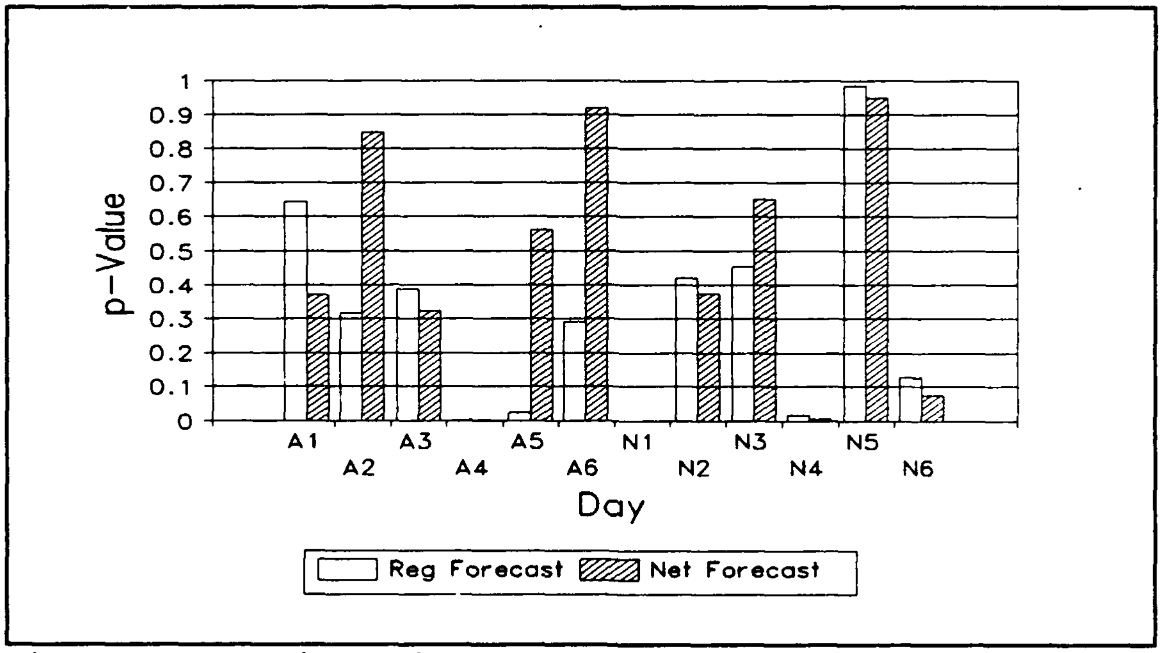


Figure 4.10 Paired Difference p-Values

3. There appears to be circumstantial evidence to substantiate the hypothesis that networks trained to a lower tolerance level (or, equivalently, to a larger number of training iterations) do less well in forecasting. Networks requiring fewer training iterations appear to possess larger positive forecast correlations and smaller forecast errors, in some cases. With respect to statistically significant differences, no clear pattern emerges in relation to the two other principal analysis factors. Within this context, the reader is cautioned to accept the evidence of overtraining provisionally; many networks require thousands of training iterations, and further analysis is required to determine the extent to which networks can forecast the test sets used herein more accurately when trained to lower tolerance levels.

4. In terms of practical significance, neither meta-model set provides sufficiently acceptable forecasts to be considered for real-world implementation. Additional research is required for both modeling techniques, although this study provides sufficient insight into their behavior to be useful for further research. Accordingly, a discussion of precisely how the research performed in this study answers and supports the research objectives stated in *1.5*, together with issues pinpointed for future research, jointly comprise Chapter 5, the next and final chapter of this study.

V. Conclusions and Recommendations

5.1 Conclusions

5.1.1 Prediction. This research assessed the predictive capability of twelve backpropagation neural network metamodels *via* a baseline of an equal number of fully specified regression metamodels. In general, the networks did not approach the accuracy in prediction exhibited by their regression counterparts. While the result for this research objective is clearly negative, the overall performance and behavior exhibited by the networks is far from trivial: networks closely approximated the behavior of the regression equations for most days under study, particularly for days possessing well-behaved response variation.

By standards of common neural network research practice, the constraints placed upon the networks were severe. While imposed intentionally, the results suggest that additional representation of variance is needed for improving overall network forecasting accuracy. Neither the networks nor the regression metamodels contained a term for blocking, which appears important for adequately modeling TSAR/TSARINA (simulated) samples regardless of metamodeling method. To this end, the networks appear to suffer (considerably) from a lack of adequate problem *representation*.

Further, in mainstream neural network research, many samples are often used in developing an application. In this study, only training and validation samples were employed, in the effort to "freeze" the results and analyze network performance characteristics for a preliminary research assessment. Hecht-Nielsen recommends using three independent samples: a training set (used here), a *training test set* (not used here), and an independent test set (also used here) for training backpropagation neural networks (1990:115-119). The training test set is used in conjunction with the training set to fit the network in question, while the latter consists of unseen cases used for evaluating a network only *after* training is completed. In this way, the researcher ensures that the networks are familiar with a relatively large set of cases, and that the network is adequately generalizing (predicting) from the training set.

A clear implication is that samples used for network training must truly sample important observations--while the neural network literature seems abundant with warnings regarding the necessity for employing large samples for adequate representation, the real issue seems more analogous to that of importance sampling used in traditional simulation modeling and analysis. In essence, it appears to be a quality over quantity issue, although networks indeed seem to require considerably more data than traditional modeling techniques.

5.1.2 Fitting versus Prediction. With respect to model fitting, networks were superior to the regression metamodels in every case; so much so, they appeared incapable of generalizing (predicting) beyond the training set (as evidenced by small positive and even negative correlations between forecasts and test set actuals). While circumstantial evidence exists to support the conclusion that some networks developed in this study were overtrained, the more probable rationale is that the networks were not constructed in a manner that adequately models the large degree of response variation present in the training and testing samples. Hecht-Nielsen notes that in monitoring training set and training test set Mean Squared Error (MSE) simultaneously, one may find that the training test set reaches its lowest MSE at a point where the training set MSE is *not* at its minimum (1990:117). In this way, the researcher ensures that backpropagation networks more optimally predict unseen cases by measuring the interaction of training and training test sets--in essence, the researcher *empirically* distinguishes goodness-of-fit from goodness-of-forecast.

5.1.3 Neural Network Properties. For the neural networks developed in this research, the problem with generalization (predicting unseen cases) appears to be primarily a function of architecture selection. While the need for the representation of blocking has already been noted, the issue of superfluous hidden nodes also requires comment: Sanger indicates that the addition of too many hidden nodes renders some of the them superfluous (1989). With respect to generalization, too many hidden units will eventually cause the network to simply "memorize" the training set (*via* overfitting) (California Scientific Software Technical Support, 1991), with poor prediction a likely consequence.

Conversely, when determining the minimum number of hidden units required by the networks developed in this research, it was discovered that using too few renders the net incapable of converging to the desired error tolerance during training. Further, a test of several networks possessing minimal architectures (8 hidden nodes or less) revealed that generalization capabilities were poorer than the 10-14-1 networks reported in this study.

In addition, the networks were extremely sensitive to changes in and the interaction of error tolerances, the number of hidden nodes, and the learning rate, as noted during the preliminary stages of this study. Although the precise relationship between the number of hidden units and learning rate settings is unclear, it appears there is a trade-off between learning rate settings and the number of hidden nodes contained in the network's architecture. While the effect on generalization remains unclear, it does appear an analysis of the generalization capabilities of networks possessing fewer hidden nodes and larger error tolerances than those developed in this research is required.

5.1.4 Other. The clear superiority of networks in fitting data suggests that they offer a powerful means of approximating functions for *known* data, which is invaluable in real-time and expert system environments requiring a "look-up" capability (the conceptual leap from an expert system to a binary-valued neural network is indeed a small one). To this end, the difficulty the networks experienced in predicting unknown values appears to be precisely the same characteristic that makes them valuable in fitting training samples.

5.2 Recommendations for Future Research

5.2.1 General. This study deals with an issue that is relevant to the current efforts of ensuring efficient and effective combat capability in the face of a constant reduction in the budgetary resources for accomplishing same. Diener's work (1989), from which this study takes its genesis, is directly engaged in identifying those resources most important for sustainability and readiness for an F-15 air base; this research focuses on predicting the number of sorties that can be flown on a given day for a given level of air base resources. Both studies attempt to engage these questions at the systems level, in lieu of focusing upon a single functional area, i.e., questions are asked of the entire system, sacrificing sub-optimal precision for systems-level understanding. Due to the criticality of the latter, especially when confronted by the reality of continually decreasing budgetary resources for defense, it is strongly recommended that the research efforts in both studies be continued.

5.2.2 Recommendations for Future Research. The prediction of sorties flown from a given level of air base resources is a research topic with both interesting theoretical implications and practical significance. Accordingly, the research has strong merit for further investigation. The following recommendations are made for further study:

1. Network Architecture. Alternative backpropagation topologies should be explored. Most notably, a means of representing the blocking variables analogous to that commonly used in regression metamodeling should be examined first. Adequate modeling representation of response variation was found to be important for network forecast accuracy, particularly for Attack Scenario days.

In addition, other types of neural networks should be investigated; see Tarr (1988) for a hybrid Kohonen-Backpropagation neural network design that may provide additional modeling capability for the problem under study in this research.

2. Time-Series Modeling. The exploration of creating two larger models that incorporate *all* No-Attack and Attack

Scenario Days as multivariate time series should be investigated. It is recommended that a time series model representing the 30 (daily) sorties flown responses be explored, as this topology more closely models the simulation experiment than the networks developed in this study, and will provide additional information with respect to correlated responses across days. Given a set of network alternatives that model the same data from different perspectives (i.e., longitudinally and cross-sectionally), significance testing between prediction results may promote more enlightened network problem representation.

3. Neural Network Weight Matrix Interpretation. Methods for interpreting neural network weight matrices are receiving much recent attention, due to the need for assessing the importance of model variables. Most notably, principal component analysis (Sanger, 1989, Hertz et al., 1991:197-210), Chernov faces (Howell, 1991), and Garson's method (Garson,1991) show promise for interpreting the internal representations formed during network training and for assessing the relative significance of model variables. In the spirit of Diener's analysis (1989), these weight interpretation methods should be explored in efforts to assess relative independent variable importance.

4. Sample Size. The sample sizes collected for modeling and testing should most likely be increased. Neural networks are clearly data-driven models, and the

predictive capabilities appear to be relatively poor unless a relatively large number of (very) representative observations predominate the training sets employed. To this end, the technique appears less robust than traditional regression methods--for this research. The reader is cautioned not to equate sample size with sample importance, however--as noted previously, size appears less of an issue than the relative importance of what is sampled, with respect to problem representation.

5. Real-valued Data or Additional Factor Levels. As a purely exploratory issue, resource factor indicator variables could be replaced by real-valued scalars where possible, to expand the modeling space and representation of the problem to the networks. Where not practicable, indicators could still be used. The result would be a mixture of real-valued and symbolic data comprising the input vector (resource policy), which is amenable to neural network application, and may promote better network generalization.

6. As an additional alternative, the simulation experiment could be expanded to more factor levels to accommodate values falling between "low" (0) and "high" (1) (for example "low", "near-average", "average", "above-average", "near-high", and "high" could possibly correspond to a fractional factorial design with factors at 6 levels). This approach, in effect, generates a "degree of membership"

approach to factor value representation analogous to that used in fuzzy-set data representations. The reader is referred to Kosko (1992) for a detailed discussion of the relationship between neural networks and fuzzy systems.

5.3 Epilogue

Simulation metamodels not only provide a unique opportunity for researchers attempting to focus deeply upon the behavior of data, they also constitute a bridge between highly trained researchers and practicing managers. In modeling several alternative Scenarios, simulation allows us to explore the effects and consequences of our actions and choices in many different situations. In contrast, the analytic modeling of physical systems essentially tells one story--the current one. *Via* simulation analysis, together with metamodeling, we may analyze and compare the systemic behaviors of several *logically possible* states of affairs in lieu of analyzing only a single *actual* state of affairs.

Much of this study was concerned with the exploration of a new technique that offers promise in deriving an analytical representation of a problem of sincere concern to managers facing continual resource reductions. In light of the practical significance of resource modeling analyses, we strongly encourage the continuation of the exploratory research presented here and in Diener (1989). The interaction of both approaches to metamodeling is rich in both theoretical and practical significance.

APPENDIX: <u>Neural Network Weight Matrices</u>

Attack Scenario Day 1 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	0.013	-0.997	2.078	-3.043	0.468	0.419	1.707	1.882	1.583	1.134	-6.878
2	-8.000	-0.645	0.473	-0.704	-1.433	-1.156	-0.708	-1.668	0.997	-0.864	0.491
3	1.087	0.471	-1.836	1.781	-5.595	-0.385	-0.169	3.543	1.033	-0.091	-0.275
4	5.635	-8.000	-8.000	-8.000	-2.155	4.253	-8.000	-8.000	-0.232	-8.000	-1.696
5	2.453	2.445	-3.053	-2.132	3.917	-0.085	-4.343	-2.949	2.429	0.717	0.571
6	1.571	-0.974	1.399	-0.995	-7.809	3.945	-4.656	2.494	-1.071	-1.708	5.011
7	5.843	2.838	-0.188	0.207	3.730	-1.580	0.152	-2.897	-1.451	-5.213	-1.383
8	-8.000	4.622	2.637	-1.619	2.994	-2.658	2.037	4.888	-7.945	-4.560	-2.258
9	0.572	-2.641	0.047	-2.181	1.165	1.443	-0.319	5.212	1.428	-2.704	-5.199
10	5.501	0.898	2.464	-2.121	-0.913	0.680	0.451	-4.259	-2.539	-4.783	-0.052
11	4.000	0.049	1.059	-4.717	-2.180	2.519	-1.222	-1.427	-2.392	1.528	-0.014
12	-8.000	-6.192	-7.253	-7.707	-5.072	-5.783	-4.068	-4.548	-6.110	-4.836	-7.444
13	-2.346	3.020	-0.898	-1.930	0.868	7.988	-4.271	-7.062	7.999	-3.113	-4.490
14	3.497	-5.117	-2.547	1.356	-0.097	2.827	-1.298	0.006	-3.979	-1.999	1.155

H\O	1	2	3	4	5	6	7	8	9	10	11
	-4.276	-3.699	-3.369	2.030	-2.550	4.108	3.709	-2.626	-1.242	-4.099	-1.905

H\O	12	13	14	Bias
	-8.000	-0.927	-1.279	2.068

Note: H\O is a row matrix (1 output node)

Attack Scenario Day 2 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-2.729	-5.530	4.128	-6.893	0.726	0.396	-0.596	7.925	0.430	1.744	-7.908
2	2.251	-3.142	0.971	3.270	-1.967	1.705	-1.991	-2.333	-3.410	-2.527	-0.991
3	3.433	1.413	2.509	-0.639	-2.895	-2.204	2.636	-2.161	-2.876	1.321	-0.390
4	0.809	7.977	-2.740	-8.000	2.714	-6.952	2.956	-7.901	2.827	-1.420	-1.726
5	-5.041	-7.906	-0.079	0.339	-4.626	0.256	-7.998	6.544	-7.341	1.019	-7.751
6	-1.610	-5.191	-0.846	0.368	-6.028	2.479	4.076	-3.543	-4.412	-0.383	7.423
7	7.314	1.035	-4.704	-0.692	0.225	-0.914	-1.633	-7.987	-2.737	-8.000	2.605
8	-8.000	7.889	1.717	-1.736	7.964	6.280	-7.891	0.673	2.371	-5.796	-1.824
9	-1.195	0.565	-8.000	-5.741	1.961	-1.563	0.825	1.941	-7.141	-0.408	1.118
10	-0.246	3.515	0.905	-4.562	-1.061	-1.865	-0.625	-4.264	1.439	-1.382	-0.572
11	0.055	-2.562	-6.420	-7.596	-1.626	2.080	4.360	-2.811	7.997	0.038	0.801
12	1.152	-7.122	-7.317	-7.802	-7.593	-6.687	5.079	-7.975	-1.545	-7.415	-4.167
13	0.593	1.179	2.821	-8.000	-0.896	-0.660	0.471	2.095	1.521	2.361	7.504
14	7.817	-0.001	4.624	-3.413	-1.168	-7.593	0.763	0.206	-2.866	3.783	-1.853

H\O	1	2	3	4	5	6	7	8	9	10	11
	-5.372	-2.808	-4.393	2.932	-8.000	4.625	2.444	2.163	-6.732	-7.999	-2.705

H\O	12	13	14	Bias
	5.639	5.778	3.576	-6.228

Note: H\O is a row matrix (1 output node)

Attack Scenario Day 3 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-0.972	-2.423	1.559	-1.070	-1.451	0.250	1.322	2.127	0.343	1.425	-5.022
2	2.467	-0.801	2.618	2.406	-0.071	0.840	0.872	-4.660	-0.380	-2.111	-2.987
3	-0.871	-0.264	1.396	-2.632	-4.827	-1.046	-1.510	1.730	-0.187	-0.259	-0.130
4	-2.488	-0.163	-1.833	-2.058	4.837	0.056	2.684	-0.995	0.726	0.117	-6.666
5	-3.795	-0.931	-2.712	-2.027	4.401	1.820	0.987	-1.411	-0.906	2.132	-1.155
6	-4.851	0.583	0.023	0.938	-0.128	4.665	1.385	-2.444	2.559	2.062	-2.447
7	6.117	0.629	-2.138	0.179	1.006	-0.205	0.563	-2.123	-0.971	-4.839	-1.307
8	-3.444	1.390	1.903	-1.758	-0.561	-0.985	-3.342	5.860	-1.867	-2.242	-3.440
9	2.790	-2.779	0.987	-2.589	2.757	-0.738	0.262	3.407	-2.265	-4.214	-4.165
10	1.986	-0.967	-2.040	-1.054	-3.695	0.031	0.132	-2.599	-1.027	-1.622	0.504
11	1.797	3.887	-2.375	-3.802	2.413	0.926	2.642	-8.000	2.292	-1.287	-5.083
12	1.171	3.884	0.005	3.550	-1.759	0.254	-1.374	2.736	1.535	-2.700	-6.343
13	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000
14	2.566	-4.127	3.517	-2.165	-3.460	-3.336	0.642	-0.455	-1.528	-0.664	1.719

H\O	1	2	3	4	5	6	7	8	9	10	11
	-7.991	-4.306	-6.164	5.715	-4.104	4.711	7.288	5.152	-7.136	-7.999	-4.866

H\O	12	13	14	Bias
	-3.564	0.082	4.928	-0.047

Note: H\O is a row matrix (1 output node)

Attack Scenario Day 4 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	1.422	-4.061	2.010	1.199	-1.477	-1.149	1.307	1.677	1.115	1.216	-3.232
2	1.154	-0.465	-2.188	-4.685	-4.075	-8.000	2.473	-1.450	3.966	2.800	-1.433
3	-2.707	4.373	-2.482	-5.951	-8.000	0.522	-0.405	2.574	4.677	-3.078	-1.345
4	-0.270	-1.779	-0.184	0.131	-1.704	1.735	2.066	-0.955	-0.522	2.883	0.225
5	-0.438	0.519	0.213	-0.675	2.988	3.104	-0.174	-0.740	2.196	1.250	0.705
6	0.882	-3.513	0.338	4.627	-5.197	1.288	-1.699	-0.971	2.383	-0.877	5.705
7	4.363	-0.505	-0.602	-0.183	4.402	-1.305	-0.538	-2.078	-0.404	-2.981	-0.121
8	-3.956	1.670	-1.595	-3.306	3.155	-0.811	-1.046	3.287	0.489	-3.367	0.614
9	-1.937	-1.305	-1.086	-1.728	0.246	-1.793	-0.820	0.367	-2.397	-1.146	0.154
10	5.845	1.604	-1.162	-0.840	0.777	-1.584	-3.603	-2.683	0.758	0.363	-0.521
11	-0.054	2.896	0.334	-1.289	-1.710	2.115	0.819	-2.444	-1.874	0.937	-4.098
12	0.760	-0.936	-1.448	-0.294	-1.237	2.364	-4.268	0.424	1.358	2.284	-3.567
13	-2.078	5.342	-1.198	1.434	2.142	6.589	-2.147	-2.728	0.470	3.276	-3.431
14	2.537	-0.062	-0.052	-0.676	-0.398	-0.713	-1.793	-5.042	-1.829	-1.067	-1.276

H\O	1	2	3	4	5	6	7	8	9	10	11
	-2.985	-2.278	-1.467	2.315	1.990	1.652	-2.115	1.729	-6.438	2.803	-2.435

H\O	12	13	14	Bias
	-2.646	-2.491	-3.298	-1.102

Note: H\O is a row matrix (1 output node)

Attack Scenario Day 5 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	0.767	-5.395	1.630	-1.940	0.047	-0.040	0.760	2.446	0.050	3.328	-4.175
2	0.739	1.245	4.461	-8.000	3.580	-4.576	3.559	-7.106	-1.093	-8.000	-1.103
3	-1.149	2.883	-0.494	-4.370	-1.861	-1.903	-0.625	-2.870	0.798	1.670	-0.445
4	2.028	-4.818	-1.381	0.233	-1.654	5.433	-0.585	0.102	-0.185	2.413	-1.699
5	0.363	1.802	-5.830	-2.426	1.542	3.836	-1.490	-2.083	-2.724	3.640	1.156
6	0.649	-5.071	-0.020	0.770	-3.396	1.869	-2.153	0.750	1.145	-5.239	3.184
7	3.129	-1.194	2.227	-0.316	4.653	-0.533	1.021	-3.635	-1.171	-5.551	0.792
8	-2.914	1.818	1.652	0.134	3.732	-1.463	1.946	2.048	-1.879	-3.305	-1.075
9	6.976	-1.156	1.840	-3.751	7.661	-7.921	-5.459	-3.711	-8.000	-6.934	-6.345
10	5.544	-0.942	-1.659	-0.635	-2.317	-0.352	-6.034	-5.598	2.276	-0.115	4.185
11	-0.451	1.060	-0.269	-4.430	-2.316	0.241	-0.920	-2.796	-4.132	-0.407	-0.121
12	-0.946	1.610	-2.594	-0.088	-6.431	1.938	-2.369	3.112	1.169	5.979	-2.304
13	-2.851	-1.070	2.649	1.378	3.849	2.486	-7.911	-4.587	2.859	4.334	-4.138
14	3.315	-3.517	0.341	-5.016	5.329	-8.000	-7.873	-4.586	-4.054	-2.767	-8.000

H\O	1	2	3	4	5	6	7	8	9	10	11
	-2.317	-0.733	2.289	4.187	-1.695	-2.318	-3.967	3.775	1.174	3.129	-3.240

H\O	12	13	14	Bias
	-3.134	-1.354	-2.389	0.781

Note: H\O is a row matrix (1 output node)

Attack Scenario Day 6 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	0.219	-4.659	3.817	5.358	1.727	-3.134	5.560	4.851	2.043	1.903	-5.494
2	0.299	3.249	-3.474	-5.081	-8.000	-7.217	2.778	0.672	-4.573	-4.730	0.315
3	4.614	4.515	-2.598	-5.782	-3.258	-6.674	-4.042	1.135	-2.846	1.194	7.510
4	7.999	7.999	-4.409	7.999	7.999	7.999	7.999	7.999	-4.409	7.999	7.999
5	-1.766	-2.644	-7.965	5.053	6.563	7.439	-7.351	5.542	0.533	7.259	-0.811
6	-0.917	-2.417	4.076	4.487	0.829	-3.282	5.057	4.071	1.665	0.757	-3.224
7	-8.000	-4.661	-6.649	1.088	6.695	3.605	-7.104	-3.194	-6.220	0.394	-6.647
8	-7.837	6.653	1.179	-1.026	0.241	7.012	0.333	-1.470	-1.224	-6.278	7.999
9	-0.862	1.324	1.696	0.765	-1.912	-0.349	0.889	1.664	-1.336	-1.865	-2.108
10	6.333	-2.399	0.059	-2.168	2.234	-1.645	1.526	-6.893	4.267	4.174	-1.864
11	-1.536	7.365	-3.353	7.999	-8.000	-4.893	-0.264	-6.208	-3.089	-5.393	-7.966
12	-7.945	7.999	7.658	0.811	-8.000	-7.876	-6.999	-2.147	-2.188	-7.101	1.594
13	3.137	4.715	2.337	-8.000	-7.118	7.541	-2.440	-4.044	4.915	4.628	-7.495
14	7.735	7.115	7.795	-4.745	0.952	-3.350	-8.000	-5.865	-4.726	-4.873	-8.000

H\O	1	2	3	4	5	6	7	8	9	10	11
	-7.061	-4.173	1.772	0.369	0.981	7.127	1.288	1.650	3.471	1.888	-3.630

H\O	12	13	14	Bias
	-2.226	-1.186	-1.371	-3.516

Note: H\O is a row matrix (1 output node)

No-Attack Scenario Day 1 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-4.151	-4.728	3.218	-2.403	-0.492	-0.541	-0.915	0.393	-0.545	-2.391	-0.509
2	-5.604	-2.163	1.278	-2.943	3.064	-3.614	-0.990	-7.325	-0.186	-5.035	-1.124
3	1.820	3.608	0.370	-3.636	-3.636	-1.820	3.045	-0.126	-1.128	3.475	-2.404
4	-1.224	2.246	-7.908	0.745	1.153	0.999	0.009	-7.030	2.993	4.801	-0.736
5	-2.419	3.435	-1.448	-6.341	3.861	2.953	-3.533	2.304	3.870	4.304	-0.040
6	-4.729	0.063	2.188	2.002	-5.891	-1.142	0.595	4.007	1.131	-4.947	3.135
7	5.000	-0.543	3.264	-1.438	2.992	-0.044	-1.252	-2.325	-1.165	-2.964	-3.318
8	-3.874	1.392	5.905	-2.528	-4.508	3.170	-5.446	5.641	-8.000	-7.264	-3.882
9	1.548	-0.349	-4.776	-7.893	-1.104	4.273	6.021	3.284	-3.047	-2.766	-7.784
10	2.230	1.775	-7.776	-8.000	0.370	-4.881	-5.708	-4.923	-3.676	0.382	-0.819
11	-4.496	7.646	5.176	-7.390	-4.956	1.265	1.543	-4.235	2.223	-2.653	-3.924
12	0.752	6.433	-2.547	-0.284	-4.217	0.171	2.761	-2.840	5.477	2.910	-2.458
13	-2.623	3.872	-1.601	-2.030	5.866	4.808	-1.386	2.527	2.797	5.281	-4.723
14	7.998	-3.260	2.001	-1.973	0.487	-3.177	-3.643	-1.802	-2.282	-3.428	-0.777

H\O	1	2	3	4	5	6	7	8	9	10	11
	-4.782	-5.866	1.786	2.282	1.976	2.282	2.789	-1.503	-1.721	-3.663	-1.072

H\O	12	13	14	Bias
	-3.501	-2.297	-2.680	0.305

Note: H\O is a row matrix (1 output node)

No-Attack Scenario Day 2 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-2.342	-2.106	0.607	0.105	1.356	1.635	-0.869	0.917	-4.027	6.048	-5.163
2	-1.599	-1.581	-4.101	-1.261	0.181	-4.351	-3.924	-1.945	-0.111	-1.538	1.013
3	2.628	7.635	3.160	0.290	-1.405	-7.707	-3.439	1.275	-2.104	1.263	-1.920
4	-1.546	-2.319	0.559	-1.126	0.303	-0.925	2.238	-0.408	0.358	1.839	4.360
5	-3.179	0.083	-1.681	2.838	7.980	5.435	-0.588	3.205	7.999	-1.035	2.822
6	0.862	-1.392	0.801	-1.713	-0.894	1.163	1.514	3.071	-2.285	-4.641	0.780
7	5.579	1.619	-0.511	-1.688	-0.537	-4.054	-0.087	3.535	-0.805	-1.469	-1.117
8	-2.267	-1.239	4.760	-0.300	5.465	0.491	-2.916	3.679	-2.269	-1.740	-1.476
9	3.742	1.370	-2.040	-2.645	-0.723	-0.149	0.160	-4.069	1.708	-1.815	-3.003
10	5.489	3.148	-4.815	-7.103	-6.351	-0.911	-0.555	-5.487	2.388	-3.523	-0.458
11	3.612	0.468	-0.932	-6.265	-1.414	-2.394	1.851	-2.219	3.083	-0.005	1.307
12	3.390	-0.249	1.798	4.374	-8.000	-3.265	-2.438	5.503	4.562	2.957	-5.630
13	-8.000	-8.000	-8.000	-8.000	-8.000	-8.000	-0.257	-8.000	-8.000	-8.000	-8.000
14	-1.528	-2.448	-0.660	1.103	-0.140	-1.351	-1.111	-1.038	-0.585	-7.685	-0.226

H\O	1	2	3	4	5	6	7	8	9	10	11
	-1.692	-3.322	-1.486	2.511	-2.411	-2.060	2.416	1.695	-2.312	3.282	-1.420

H\O	12	13	14	Bias
	-1.130	-0.096	-4.391	1.867

Note: H\O is a row matrix (1 output node)

No-Attack Scenario Day 3 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-1.022	-3.670	0.455	1.460	1.577	1.175	-2.559	-1.979	-1.356	1.016	-1.902
2	-0.617	-8.000	4.200	-6.105	-1.862	-3.033	-1.094	-8.000	-0.732	-0.257	-0.725
3	2.484	3.678	-1.972	-1.720	-1.200	-0.723	0.798	0.482	0.727	1.020	0.789
4	-0.251	-0.134	-0.804	-2.498	-1.928	3.012	1.987	-1.380	1.175	2.542	-0.215
5	-2.566	-0.386	2.051	-4.813	7.674	2.786	0.044	-0.617	3.052	1.758	2.400
6	-2.281	-2.094	0.144	1.259	-5.429	3.280	0.993	1.019	-0.651	-1.624	1.156
7	4.212	-4.300	-4.884	-4.771	-2.349	-4.820	-6.672	4.369	-0.682	-0.529	2.067
8	-2.114	1.332	1.308	-1.417	1.496	-1.568	0.013	-0.135	-2.205	-3.002	2.526
9	0.415	-2.210	-1.418	-2.308	2.542	0.999	0.005	-1.101	-2.069	-3.584	-2.558
10	5.189	2.193	-7.352	-3.165	-4.630	-0.004	-4.948	-8.000	1.431	1.614	-2.503
11	-0.573	1.361	0.590	-4.189	-2.947	0.661	-1.850	-1.989	-0.341	1.594	1.425
12	-0.493	2.634	-0.794	2.005	-2.150	1.176	-2.851	2.431	2.061	2.447	-4.279
13	-2.880	3.805	0.725	-1.299	3.663	2.177	-3.745	-1.254	4.056	1.691	-2.661
14	2.346	-2.580	-2.331	-0.589	-0.441	-0.195	-2.484	-2.138	-0.594	-1.343	1.470

H\O	1	2	3	4	5	6	7	8	9	10	11
	-1.720	-2.519	-2.349	2.034	1.917	-2.177	-1.456	2.570	-4.333	-1.941	-1.613

H\O	12	13	14	Bias
	2.927	-1.297	4.209	-0.552

Note: H\O is a row matrix (1 output node)

No-Attack Scenario Day 4 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-5.968	-3.354	3.126	-3.798	1.499	0.760	-0.661	4.292	2.770	2.540	-3.067
2	-3.175	-2.786	0.921	-2.277	0.837	-1.646	1.588	-5.259	1.211	-1.539	-1.106
3	-4.608	-7.982	-7.991	-7.951	-2.962	-8.000	-1.271	-7.704	-7.222	-6.796	-7.999
4	-1.533	-2.748	1.037	-2.298	-0.184	2.842	3.981	-5.078	1.523	-0.761	0.239
5	4.360	0.160	0.070	-3.959	2.688	1.739	-5.138	-1.014	6.402	5.915	-1.697
6	-8.000	-8.000	-0.796	-1.108	-5.379	1.646	0.178	1.422	-2.855	-3.950	3.963
7	7.941	-8.000	0.654	-3.348	7.413	0.866	0.402	-5.320	3.568	-7.070	-2.602
8	-4.559	6.552	2.048	1.192	2.464	-4.499	-0.851	4.349	-2.998	-5.089	-2.750
9	0.056	-1.627	-0.318	-0.979	1.625	0.515	3.447	0.741	-1.042	-3.405	-3.541
10	7.999	2.219	-3.085	-3.500	3.314	-1.158	-0.227	-2.577	-3.115	0.305	-1.402
11	-7.631	0.016	0.255	-0.612	-1.383	1.360	-1.770	-0.184	-0.659	-1.111	0.473
12	0.446	-0.914	-0.156	-1.206	-1.239	2.398	0.190	-2.116	2.242	2.067	-4.772
13	-6.126	2.558	-1.688	-2.042	4.061	4.450	-4.314	2.893	1.401	3.541	-2.534
14	7.044	3.020	-1.318	-2.685	4.465	-1.245	-2.168	-4.210	-3.515	-0.470	-0.775

H\O	1	2	3	4	5	6	7	8	9	10	11
	-1.961	-2.218	7.290	1.894	1.359	1.680	1.178	1.672	-2.052	2.510	-2.277

H\O	12	13	14	Bias
	-3.672	1.969	-3.382	-1.333

Note: H\O is a row matrix (1 output node)

No-Attack Scenario Day 5 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-0.544	0.047	2.393	-2.241	-0.093	0.321	2.210	0.610	-0.140	-1.184	-1.333
2	-0.453	-2.666	-1.920	1.425	0.169	-0.095	-5.398	-2.238	0.584	-3.592	-0.462
3	-7.865	-5.469	-7.116	-5.142	-4.792	1.722	0.841	-0.640	-6.637	-0.950	-3.5[illegible]
4	-1.584	-0.688	2.314	-1.199	-2.552	0.984	-1.691	0.322	1.144	3.129	-1.8[illegible]
5	0.321	3.420	3.040	-2.663	1.583	2.678	-1.182	-0.764	-0.735	4.700	2.099
6	-3.597	-4.040	-0.231	5.013	-7.363	5.041	-3.424	5.642	-0.968	-3.077	-0.681
7	5.292	-0.703	1.029	-3.380	-0.414	-0.785	3.102	-0.029	1.566	-6.677	3.588
8	-1.565	3.317	0.358	-2.828	0.717	0.359	-1.215	2.785	-0.943	-1.639	1.330
9	-5.315	-4.675	-4.602	1.033	6.221	-2.524	0.670	0.141	-2.320	-5.373	-3.560
10	7.177	2.663	-2.640	-4.233	0.866	-0.953	2.287	-4.231	-0.084	-0.392	-3.937
11	1.834	3.714	-4.795	-8.000	-3.607	2.622	-3.255	-1.160	-1.758	0.753	0.603
12	-1.326	-2.421	0.497	-2.488	-5.528	-1.622	-3.376	0.118	4.709	4.883	-0.609
13	0.235	4.899	0.973	-2.495	2.341	1.994	-0.625	-1.270	1.763	2.557	-3.246
14	1.998	-0.577	0.975	0.260	-0.948	-1.346	-1.845	-0.205	-3.055	-0.027	0.873

H\O	1	2	3	4	5	6	7	8	9	10	11
	-2.335	-2.206	-2.515	2.628	2.096	-1.616	0.953	1.970	-1.882	0.408	-1.268

H\O	12	13	14	Bias
	-1.948	-1.881	-1.710	-0.236

Note: H\O is a row matrix (1 output node)

No-Attack Scenario Day 6 Weight Matrices
(Input to Hidden, Hidden to Output)

H\I	1	2	3	4	5	6	7	8	9	10	Bias
1	-5.474	0.603	-2.275	3.289	2.580	-7.945	4.856	-2.966	0.963	1.070	-2.806
2	-1.307	-7.180	4.150	-6.694	-0.617	-6.816	-8.000	-8.000	7.930	-0.225	-3.599
3	2.037	2.600	-5.408	-3.039	6.804	4.465	0.542	3.294	-4.740	3.569	-2.296
4	-3.289	-2.429	-2.411	-0.303	1.803	6.523	2.063	0.401	1.237	4.026	-0.871
5	6.008	2.943	-0.941	-1.189	4.972	0.871	-8.000	-1.605	-3.033	7.558	-1.873
6	-0.196	-0.568	2.103	1.126	-1.278	0.917	-4.138	-0.653	0.104	-3.166	2.788
7	7.932	6.584	6.816	6.548	-0.011	-0.125	7.929	-0.727	0.429	-7.406	1.944
8	-4.258	6.241	3.792	-2.125	4.939	0.083	-5.469	-7.573	-2.187	-4.181	0.732
9	1.430	4.669	-5.573	4.694	4.749	-7.182	-2.763	-7.092	1.041	-3.567	1.692
10	2.790	0.544	-0.079	0.368	-2.371	-3.847	-1.879	-2.649	-2.750	-1.440	3.296
11	1.719	5.107	-7.913	-0.966	-3.753	-3.669	-1.022	-6.484	-4.814	4.067	-2.453
12	-1.231	0.281	-0.090	0.343	2.635	-0.954	-2.665	4.114	0.800	-0.749	-1.025
13	-2.491	-0.280	-0.231	2.261	1.545	6.775	-1.518	0.301	-0.636	4.986	-2.845
14	4.395	-0.410	-1.910	-0.952	-0.978	-4.292	-2.429	-0.696	-2.813	0.011	1.636

H\O	1	2	3	4	5	6	7	8	9	10	11
	-1.585	-0.973	-0.844	1.540	0.928	-1.643	3.114	0.683	0.612	1.865	-1.585

H\O	12	13	14	Bias
	1.541	-1.290	-2.075	-3.468

Note: H\O is a row matrix (1 output node)

Bibliography

Armstrong, J. Scott. Long Range Forecasting, From Crystal Ball to Computer. New York: John Wiley & Sons, 1978.

Barron R. L., Gilstrap, L. O., and Shrier, S. "Polynomial al and Neural Networks: Analogies and Engineering Applications," Proceedings of the International Conference on Neural Networks, II: 431-493. New York: IEEE Press, 1987.

Baum, Eric B. "A Proposal for More Powerful Learning Algorithms," Neural Computation, 1: 201-207 (September 1988).

Benson, P. George, PhD, Professor. Telephone Interview. Curtis L. Carlson School of Management, University of Minnesota, Minneapolis MN, 25 July 1991.

California Scientific Software. Brainmaker Professional Users Guide and Reference Manual (Second Edition). Grass Valley CA: 1990.

California Scientific Software Technical Support. Telephone Interviews. California Scientific Software, Grass Valley CA, March through July 1991. (Names witheld by request).

Cowan, Jack D. and David R. Sharp. "Neural Nets and Artificial Intelligence," DAEDALUS, 117: 85-121 (Winter 1988).

Diener, David Alan. Forecasting Air Base Operability In A Hostile Environment: Estimating Metamodels From Large-Scale Simulations. Unpublished PhD dissertation. Purdue University, West Lafayette IN, 1989.

Fishwick, Paul A. "Neural Network Models in Simulation: A Comparison with Traditional Modeling Approaches," Proceedings of the 1989 Winter Simulation Conference, 702-710.

Friedman, Linda Weiser. The Design and Analysis of Multiple Response Simular Experiments. Unpublished PhD dissertation. The Polytechnic Institute of New York, New York, 1983.

Garson, G. David. "A Comparison of Neural Network and Expert Systems Algorithms with Common Multivariate Procedures for Analysis of Social Science Data", to appear in Social Science Computer Review, 9 (Fall 1991).

Hecht-Nielsen, Robert. Neurocomputing. New York: Addison-Wesley Publishing Company, Inc., 1990.

_____. "Neurocomputing: Picking the Human Brain," IEEE Spectrum: 36-41 (March 1988).

Hertz, John, Anders Krogh, and Richard G. Palmer. Introduction to the Theory of Neural Computation. Redwood City CA: Addison-Wesley Publishing Company, Inc., 1991.

Hinton, Geoffrey E. "Connectionist Learning Procedures," Artificial Intelligence, 40: 185-233 (1989).

Howell, Jim. "Inside a Neural Network", AI Expert, 29-33 (November 1990).

Khanna, Tarun. Foundations of Neural Networks. New York: Addison-Wesley Publishing Company, Inc., 1990.

Kosoko, Bart. Neural Networks and Fuzzy Systems. Englewood Cliffs NJ: Prentice Hall, Inc., 1992.

Lapedes, Alan and Robert Farber. How Neural Nets Work. Los Alamos National Laboratory Technical Report Number LA-UR-88-418, Theoretical Division, Los Alamos National Laboratory, Los Alamos NM. (Presented at the Proceedings of the IEEE, Denver Conference on Neural Nets, 1987).

Law, Averil M. and W. David Kelton. Simulation Modeling and Analysis (Second Edition). New York: McGraw-Hill Book Company, 1991.

_____. Simulation Modeling and Analysis (Second Edition). New York: McGraw-Hill Book Company, 1982.

Lawerence, Jeannete and Mark Lawerence. Brainmaker Professional User's Guide and Reference Manual (Second Edition). Grass Valley CA: California Scientific Software, 1990.

Lippmann, Richard P. "Pattern Classification Using Neural Networks," IEEE Communications Magazine, 27: 47-62 (November 1989).

_____. "An Introduction to Computing with Neural Nets," IEEE Acoustic Speech Signal Processing, 4: 4-22 (April 1987).

Makridakis, Wheelright and McGee. Forecasting: Methods and Applications (Second Edition). New York: John Wiley & Sons, 1983.

McClave, J. and Benson G. Statistics for Business and Economics (Fourth Edition). San Francisco: Dellen Publishing Company, 1988.

McClelland, James L. and David E. Rumelhart. Explorations in Parallel Distributed Processing. Boston: The MIT Press, 1988.

McCoy, Tidal. "Task One: Air Base Operability," Armed Forces Journal International, 52-56, September 1987.

McLean and Anderson. Applied Factorial and Fractional Designs. New York: Marcel Dekker, Inc., 1984.

Melendez, Kenneth, PhD, Deputy Director. Personal Interviews. Center for Artificial Intelligence Applications (CAIA), a Division of the Miami Valley Research Institute, Dayton OH, 20 June and 8 July 1991.

Meyers, Ronald H. Classical and Modern Regression with Applications. Boston: PWS Publishers, 1986.

Montgomery, Douglas C. Statistical Quality Control (Second Edition). New York: John Wiley & Sons, 1991.

Moore, Kevin. Forecasting Air Force Logistics Command Second Destination Transportation: An Application of Multiple Regression Analysis and Neural Networks. MS Thesis, AFIT/GLM/LSM/90S-37. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1990 (AD-A229629).

Morgan, M. G. and Max Henrion. Uncertainty. Cambridge UK: Cambridge University Press, 1990.

Orris, J. B. and H. R. Feeser. "Using Neural Networks for Exploratory Data Analysis," An Invited Paper Presented at the 1990 ORSA/TIMS Joint Conference, 1-8. Philadelphia: 1990A.

_____. "Using Neural Networks For Regression Analysis - Suggestions for Future Research," An Invited Paper Presented at the 1990 ORSA/TIMS Joint Conference, 1-9. Philadelphia: 1990B.

Pao, Yoh-Han. Adaptive Pattern Recognition and Neural Networks. New York: McGraw-Hill Book Co., 1981.

Pineda, Fernando J. "Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation," Neural Computation, 1: 161-172 (April 1989).

Rich, Micheal, I. K. Cohen, and R. A. Pyles. Recent Progress in Assessing the Readiness and Sustainability of Combat Forces. A Project AIR FORCE report prepared for the United States Air Force. Rand Report Number R-3475-AF. Santa Monica CA: The Rand Corporation, October 1987.

Rogers, Steven K., Matthew Kabrisky, Dennis W. Ruck, and Gregory L. Tarr. An Introduction to Biological and Artificial Neural Networks. Class text for EEN-620, Pattern Recognition I. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1990.

Sanger, Dennis. "Contribution Analysis: A Technique for Assigning Responsibilities to Hidden Units in Connectionist Networks", Connection Science, 1: 115-138 (1989).

Schwartz, Jacob T. "The New Connectionism: Developing Relationships Between Neuroscience and Artificial Intelligence," DAEDALUS, 117: 123-141 (Winter 1988).

Siegel, Sidney and N. John Castellan, Jr. Nonparametric Statistics for the Behavioral Sciences (Second Edition). New York: McGraw-Hill, Inc., 1988.

Stanley, Jeanette. Introduction to Neural Networks (Third Edition). Sierra Madre CA: California Scientific Software, 1990.

Tarr, Gregory L. Dynamic Analysis of Feedforward Neural Networks Using Simulated and Measured Data. MS Thesis, ENG 88D-54. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988 (AD-A202573).

Wasserman, Phillip D. Neural Computing Theory and Practice. New York: Van Nostrand Reinhold, 1989.

Weiss, Sholom M. and Casimir A. Kulikowski. Computer Systems That Learn. San Mateo CA: Morgan Kauffman Publishers, Inc., 1991.

Vita

James M. Dagg, born on 10 July 1955 in Dayton, Ohio, graduated from Fairmont East High School in 1973, and received a Bachelor of Arts in Philosophy, Summa Cum Laude, from Wright State University in 1981. Beginning government service with the Defense Logistics Agency in February 1984, he worked as a Procurement Analyst in the Directorate of Contracting and Production at Defense Electronics Supply Center (DESC) through March 1989. During the course of this assignment, he received two Sustained Superior Performance awards and a formal letter of appreciation for a value engineering submission. In April 1989, he transferred to the DESC Office of Operations Research and Economic Analysis, where he was engaged in logistics and artificial intelligence research until entering the School of Systems and Logistics, Air Force Institute of Technology, in May 1990.

Permanent Address: 1958 Farmside Drive
Kettering, OH 45420-3622

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1991	Master's Thesis

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
AN EXPLORATORY APPLICATION OF NEURAL NETWORKS TO THE SORTIE GENERATION FORECASTING PROBLEM	
6. AUTHOR(S)	
James M. Dagg, GS-12	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Air Force Institute of Technology, WPAFB OH 45433	AFIT/GLM/LSM/91S-11
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution unlimited	

13. ABSTRACT *(Maximum 200 words)* This exploratory study assesses the accuracy of backpropagation neural networks in predicting sortie generations, given pre-specified levels of air base resources. Single hidden layer networks and two-way interaction regression metamodels were fitted to simulated data previously generated by way of a fractional factorial design for ten factors at two levels, and subsequently tested (cross-validated) via an independent testing sample. It was determined that regression metamodels were generally superior in predicting unseen cases, while their network counterparts exhibited far better goodness-of-fit characteristics. The research consistently emphasizes that goodness-of-fit in no way necessarily implies goodness-of-prediction, in that different non-equivalent statistical measures are required to assess both these phenomena. In spite of their relatively poor performance in predicting the test sample used in this study, experimental results indicate that future research focused on applying neural network modeling techniques to sortie generation prediction and the identification of critical air base resources is warranted. Suggestions for more effective neuronal modeling include architectural experimentation, increasing sample sizes employed for network training, and representing the research problem in terms of a time series.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Neural Networks, Goodness-of-Fit, Sortie, Forecasting	180
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT Control Number AFIT/GLM/LSM/91S-11

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: AFIT/LSC, Wright-Patterson AFB OH 45433-6583.

1. Did this research contribute to a current research project?

a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

Man Years ________________ $________________

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3 above), what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

Name and Grade	Organization
Position or Title	Address